О. М. Огородникова

Вычислительные методы в компьютерном инжиниринге

Министерство образования и науки Российской Федерации
Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина

О. М. Огородникова

Вычислительные методы в компьютерном инжиниринге

Учебное пособие

Екатеринбург
УрФУ
2013

УДК 004.4:621.01
ББК 32.973.26-012.2+34.42
О39

Рецензенты:
кафедра технологии машиностроения и методики профессионального обучения Машиностроительного института РГППУ;
д-р техн. наук А. М. Потапов, главный научный сотрудник Института высокотемпературной электрохимии УрО РАН;
заместитель директора по научной работе Челябинского института путей сообщения В. Л. Федяев.

Огородникова, О. М.
О39 Вычислительные методы в компьютерном инжиниринге: учебное пособие / О. М. Огородникова – Екатеринбург : УрФУ, 2013. 130 с.

ISBN 978-5-7996-0816-3

В учебном пособии изложены необходимые сведения о терминологии и методах вычислительной математики, предваряющие углубленное изучение компьютерного инжиниринга в машиностроении. Рассмотрены уравнения и системы уравнений, задачи интерполяции и аппроксимации, численное решение дифференциальных уравнений в частных производных, метод конечных разностей и метод конечных элементов. Учебное пособие ориентировано на бакалавров и магистров, обучающихся по направлениям 220700 «Автоматизация технологических процессов и производств» и 221000 «Мехатроника и робототехника». Пособие будет полезно студентам и специалистам, интересующимся компьютерным инженерным анализом и такими популярными программами как ANSYS, NASTRAN, ABAQUS и др.

УДК 004.4:621.01
ББК 32.973.26-012.2+34.42

ISBN 978-5-7996-0816-3

Оглавление

Введение

«Что для одного ошибка, для другого – исходные данные»
Из законов Мерфи

Стремительное развитие вычислительной мощности компьютеров и прикладного программного обеспечения меняет привычное представление о характере инженерной работы. В связи с этим очевидной становится необходимость изменить содержание учебных курсов по проектированию технических объектов, добавив изложение компьютерных аспектов.

Расчётное обоснование технических проектов с использованием современных автоматизированных технологий базируется на ресурсоемких компьютерных системах и требует междисциплинарного программного обеспечения. Причем нарастающая популярность компьютерных программ инженерного анализа (Computer-Aided Engineering – CAE) у рядовых специалистов объясняется, в первую очередь, снижением стоимости высокопроизводительных компьютеров. Ведь для выполнения современных инженерных расчётов требуется большая вычислительная мощность. И если раньше для этого были необходимы громоздкие кластеры и дорогостоящие рабочие станции, то сейчас конструктору достаточно иметь персональный компьютер, снабженный 64-разрядным многоядерным процессором.

Немаловажную роль в развитии компьютерных вычислений играет и рост конкуренции в промышленности. Под давлением рынка предприятия стремятся к экономии средств и поэтому стараются перенести испытания образцов новой продукции из реального мира в виртуальный, получая при этом двойной выигрыш: разработка изделия ускоряется, а затраты на его производство снижаются. Конечно, инженерные расчёты с использованием компьютера не могут полностью заменить натурные испытания, но предварительная проверка изделия на компьютере позволяет сократить число явных ошибок и опытных образцов. Правда, для этого сами программы должны заслуживать полного доверия. И здесь важным фактором становится надежность используемых компьютерных методов вычислений, знание которых является обязательным для пользователей программного обеспечения CAE.

Учебное пособие сочетает изложение теории численных методов и их практическую реализацию через интерфейсы специализированных пакетов. В пособии изложены необходимые начальные сведения о терминологии и

методах вычислительной математики; рассмотрены нелинейные уравнения и системы уравнений, задачи интерполяции и аппроксимации, численное решение дифференциальных уравнений в частных производных. Главной целью являются метод конечных разностей и метод конечных элементов. Для полного понимания этих методов студентам предлагается самостоятельно написать несколько программ на Фортране. Практическая часть содержит полезные примеры решения типовых задач из инженерной практики.

Вся информация разбита на 11 тем, которые можно изучать независимо и в любом порядке. Некоторые темы представляют интерес не только в компьютерном моделировании инженерных объектов, но и при обработке результатов измерений технических параметров в экспериментальных исследованиях.

ТЕМА 1. Компьютерное и математическое моделирование в инженерных задачах

«Компьютерная программа выполняет только то,
что Вы ей приказали сделать,
а не то, что Вам просто подумалось,
чтобы она сделала»
Из законов Мерфи

1.1. Расчетные параметры в инженерных задачах

В инженерных задачах, как правило, ведется поиск параметров, распределенных по выделенной в пространстве области, а также изменение этих параметров во времени. Во многих случаях искомый параметр является функцией четырех переменных – трех пространственных координат и времени. Например, если требуется найти изменение температурного поля в конструкции с течением времени, то температура в такой постановке задачи является функцией времени и трех пространственных координат:

$$T = f(t, x, y, z) \ . \tag{1.1}$$

Анализируемые в инженерных задачах процессы содержат производные искомой функции четырёх переменных и описываются дифференциальными уравнениями. Поскольку искомая функция зависит от нескольких переменных, в решаемых дифференциальных уравнениях присутствуют ее частные производные. В инженерных задачах очень часто интересующие нас параметры вычисляются решением дифференциальных уравнений второго порядка с частными производными.

Например, дифференциальное уравнение второго порядка в частных производных, описывающее явление теплопроводности в технических системах, – уравнение теплопроводности Фурье (здесь приводится в усеченном виде для изотропных тел в одномерном случае), которое содержит частную производную первого порядка от температуры по времени и частную производную второго порядка от температуры по координате:

$$\rho C \frac{\partial T}{\partial t} = K \frac{\partial^2 T}{\partial x^2} + Q \ , \tag{1.2}$$

где ρ – плотность,

C – теплоемкость,

K – коэффициент теплопроводности,

Q – скорость генерации тепла в системе, обусловленная различными процессами (индукционный нагрев, выделение скрытой теплоты плавления, сгорание топлива и другие процессы, сопровождающиеся выделением или поглощением тепла).

Если в задаче анализируется несколько физических процессов, например теплопередача и деформация, тогда приходится решать систему дифференциальных уравнений, описывающих каждый из этих процессов.

Аналитическое решение дифференциальных уравнений и систем дифференциальных уравнений для большинства инженерных задач, сформулированных в полной постановке, невозможно. Решение в аналитическом виде находят для простых случаев.

Пример дифференциального уравнения, решаемого аналитически. Пусть дифференциальное уравнение механики содержит функцию координаты от времени $x(t)$, ее первую производную и имеет вид:

$$a_0 \frac{dx}{dt} + a_1 x = 0 \ , \qquad (a_0 \neq 0) \ . \tag{1.3}$$

Это дифференциальное уравнение можно охарактеризовать как

1) *обыкновенное* уравнение (т.е. для функции одной переменной);

2) *однородное* уравнение (т.е. с нулевой правой частью);

3) *линейное* уравнение (т.е. с постоянными коэффициентами);

4) уравнение *первого порядка* (т.е. с производной первого порядка).

Уравнение (1.3) можно преобразовать к виду $\frac{dx}{dt} = -\frac{a_1}{a_0} x$;

или разделить переменные $\frac{dx}{x} = -\frac{a_1}{a_0} dt$;

и получить аналитическое решение

$$x = x_{t=0} \cdot e^{-\frac{a_1}{a_0} t} \ . \tag{1.4}$$

Традиционно при инженерном проектировании многие задачи, описываемые дифференциальными уравнениями, решаются приближенно. Для получения приближенных решений используются эмпирические данные, обработанные статистически. Измеренные свойства материалов и результа-

ты наблюдений за разрушением конструкций обеспечивают точность полуэмпирических методов проектирования. Для удобства проектировщиков они сгруппированы в тематические справочники с огромным количеством графиков, таблиц и номограмм, по которым можно оценить служебные параметры инженерных систем. Полуэмпирические методы предполагают, что справочные данные подставляются в простые по структуре алгебраические выражения. Затем производятся вычисления с использованием любого подручного калькулятора. Такой подход к расчету конструктивных и технологических параметров в инженерной практике оправдан, поскольку инженеру нужны конкретные численные значения с небольшим количеством значащих цифр – температура заливки расплава в литейную форму, толщина стенки сосуда, предельное допустимое давление и т.п.

С появлением мощных компьютеров и разнообразных компьютерных программ для выполнения инженерных расчетов у специалистов появилась возможность решать дифференциальные уравнения дискретно. Вместе с тем актуальной становится необходимость развивать и осваивать компьютерные методы вычислений, оценивать точность полученных компьютером расчетных результатов.

1.2. Концепция моделирования инженерных объектов

Расчетное обоснование технического проекта базируется на моделировании объектов, составляющих этот проект. Обсуждая последовательность моделирования инженерных объектов и систем, следует выделить три связанных между собой модели: 1) физическую; 2) математическую; 3) расчётную. В основе компьютерного анализа инженерных систем лежат математические модели физических явлений.

Физическая модель содержит перечень основных процессов и явлений, определяющих поведение системы. Всякое явление природы бесконечно в своей сложности. Чтобы описать некоторое явление, необходимо выявить самые существенные свойства, закономерности, внутренние связи, роль отдельных характеристик этого явления. Выделив наиболее важные факторы, влияющие на происходящие процессы, можно пренебречь менее существенными свойствами анализируемой системы.

Математическая модель представляет собой систему уравнений, описывающих выделенные физической моделью существенные процессы, а также начальные и граничные условия. В общем случае математическая

модель с помощью уравнений и дополнительных условий описывает поведение технической системы на трех уровнях: 1) взаимодействие системы с внешней средой (по границе – граничные и контактные условия, в начальный момент времени – начальные условия); 2) взаимодействие между элементарными объемами внутри системы; 3) свойства отдельно взятого элементарного объема (уравнения состояния среды).

Математические модели обладают свойством общности, поскольку одни и те же математические уравнения описывают физические явления различной природы. С помощью математического моделирования решение научной или инженерной задачи сводится к решению математической задачи.

Для решения математических задач используются две основные группы методов: аналитические и численные. Аналитические методы позволяют получить решение задачи в виде формул. К сожалению, в инженерной практике это редко достижимо. Например, подавляющая часть нелинейных дифференциальных уравнений аналитическими способами не решается. Для решения таких задач разрабатываются и применяются методы приближенных вычислений или *численные методы*, позволяющие свести решение математической задачи к выполнению конечного числа арифметических действий над числами.

Расчетная модель адаптирует математическую модель непосредственно к проведению вычислений. С этой целью математическая модель представляется в форме, удобной для применения численных методов: задается последовательность вычислительных и логических операций, которые нужно произвести, чтобы найти решение в виде чисел с заданной точностью.

Если расчетную модель многократно реализовали на компьютере, изменяя параметры анализируемого объекта, то говорят о проведении вычислительного эксперимента. Вычислительный эксперимент, использующий в качестве исследовательского инструмента высокопроизводительные компьютеры, становится в настоящее время важным направлением научно-исследовательской деятельности.

Методы, с помощью которых задача преобразуется к виду, удобному для реализации на компьютере (вычислительные методы), можно разбить на несколько классов: 1) методы эквивалентных преобразований; 2) методы

аппроксимации; 3) прямые методы; 4) итерационные методы; 5) статистические методы.

Методы эквивалентных преобразований позволяют заменить исходную задачу более простой задачей, имеющей то же решение. Методы аппроксимации заменяют исходную задачу приближенной задачей с более простым математическим уравнением, например, линейной или квадратичной функцией. Прямые методы позволяют получить решение после выполнения конечного числа элементарных операций. Итерационные методы предполагают многократное выполнение однотипного набора действий. Статистические методы основаны на моделировании случайных величин и построении статистических оценок решений.

Обсудим компьютерное моделирование технической системы. Под компьютерным моделированием технических систем и устройств понимают адекватную замену исследуемого технического объекта расчетной моделью и ее последующее изучение методами вычислительной математики с привлечением современных вычислительных средств и компьютерных программ.

Компьютерное моделирование как универсальный инструмент исследования завоевывает прочные позиции в различных областях инженерной деятельности и становится важнейшим направлением в проектировании новых технических систем, в анализе свойств различных объектов, в выборе и обосновании оптимальных условий их функционирования.

Процесс компьютерного моделирования включает три этапа: 1) построение модели; 2) верификация и 3) виртуальное исследование.

Первый этап компьютерного моделирования – построение модели. Под компьютерной моделью понимают некоторый виртуальный объект, который в процессе исследования замещает объект-оригинал таким образом, что его непосредственное изучение дает новые знания об объекте-оригинале. Любая модель содержит ограничения и акцентирует внимание на некоторых выделенных свойствах объекта, важных с точки зрения исследуемых процессов. Но если компьютерная модель не учитывает принципиально важные качества объекта, то какие бы сложные и точные методы не применялись в дальнейшем для решения задачи, полученные результаты окажутся ненадежными, а в некоторых случаях – совершенно неверными. Познавательные возможности модели обуславливаются тем, что модель отражает некоторые существенные черты объекта-оригинала.

Второй этап компьютерного моделирования – верификация модели, ее проверка. Верификация заключается в сопоставлении расчётных результатов, полученных в рамках данной модели, с аналогичными экспериментальными данными. Для проверки модели составляется тестовая задача. Верифицированные модели можно распространить на другие процессы и системы подобного типа.

Третий этап компьютерного моделирования – виртуальное исследование модели. В процессе изучения свойств объекта модель выступает как самостоятельный объект исследования. Одной из форм такого исследования является проведение вычислительных экспериментов, когда согласно некоторому плану изменяются условия функционирования модели и систематизируются данные об откликах системы на вариации условий.

В зависимости от сложности модели применяются различные математические подходы. Для значительно формализованных и несложных моделей зачастую удается получить аналитическое решение, но физическая информативность исследования упрощенных моделей аналитическими методами невелика, поскольку позволяет оценить только порядок расчётных параметров и не транслируется однозначно на сложные объекты. Для комплексных и сложных моделей (например, при исследовании транспортных аварий или режимов резания в механообработке) аналитическое решение получить не удается. В этих случаях основными являются численные методы решения, требующие проведения расчётов на компьютерах.

Необходимость компьютерного моделирования определяется тем, что многие объекты непосредственно исследовать невозможно либо их исследование требует больших затрат времени и средств. Подходящими объектами для сугубо компьютерного моделирования являются, например, имплантанты и их поведение внутри человеческих органов или ресурс конструкций, функционирующих в космическом пространстве.

1.3. Моделирование нанообъектов

Компьютерные расчеты на базе математической модели эффективно применяются при исследовании нанообъектов и в разработке нанотехнологий. Одним из наиболее ярких примеров моделирования в области современных полупроводниковых нанотехнологий является работа, выполненная сотрудниками Национальной лаборатории имени Лоуренса в Беркли. Благодаря прямому численному моделированию квантового поведения ты-

сяч атомов на суперкомпьютере Seaborg были решены некоторые проблемы так называемых квантовых точек.

Поведение квантовых точек и их электронную структуру можно описать по аналогии с премьерой оперы на итальянском языке в театре, когда на сцене певцы стараются изо всех сил, а зрительские ряды наполнены публикой. Поведение актеров и зрителей в театре во многом похоже на поведение электронов квантовой точки. Во время представления актеры перемещаются по сцене, не выходя в зрительский зал, с другой стороны, зрители следят за действием со своих мест и не выходят на сцену. Сцена в этом примере моделирует нижние заполненные уровни квантовой точки, а зрительские ряды в партере и на галерке – возбужденные электронные уровни, обладающие более высокой энергией. При этом как зритель может находиться в любом ряду зала, так и электрон способен занять любой энергетический уровень квантовой точки, но не может располагаться между ними.

Традиционными и широко известными квантовыми точками являются выращенные на подложках полупроводниковые частицы GaN и коллоидные растворы нанокристаллов CdSe. В настоящий момент известно множество способов получения квантовых точек, например, их можно вырезать из тонких слоев полупроводниковых гетероструктур с помощью нанолитографии, а можно спонтанно сформировать в виде наноразмерных включений структур полупроводникового материала одного типа в матрице другого. Методом молекулярно-пучковой эпитаксии при существенном отличии параметров элементарной ячейки подложки и напыляемого слоя можно добиться роста на подложке пирамидальных квантовых точек, за исследование свойств которых академику Ж.И. Алферову была присуждена Нобелевская премия. Контролируя условия процессов синтеза, теоретически можно получать квантовые точки определенных размеров с заданными свойствами.

Использование квантовых точек – включений из сотен или тысяч атомов одного полупроводника в другом – позволяет изготовлять разнообразные полупроводниковые устройства – от небывало эффективных свето- и фотодиодов до кубитов для квантовых компьютеров и систем квантовой криптографии. Однако прогресс в их применении идет медленно. Дело в том, что многие из свойств квантовых точек до сих пор не очень понятны, поэтому технологам при их создании приходится действовать методом

проб и ошибок. А теоретики часто бывают бессильны, поскольку известные квантовые методы, хорошо работающие для отдельного атома или для кристаллической решетки, оказываются малопригодны, когда структура включает несколько сотен атомов.

Некоторые задачи о поведении квантовых точек в итоге были решены методом математического моделирования. Проведенные на суперкомпьютере расчеты прекрасно описали хорошо известные эффекты, а также помогли получить новую информацию об объекте. В частности, построенная модель предполагает, что локальные диэлектрические свойства внутри квантовой точки точно такие же, как и в сплошном материале. А все изменения свойств полупроводников тесно связаны с квантовыми эффектами на поверхности квантовой точки. Построенная модель после математического исследования и верификации стала мощным инструментом для развития новых технологий в изготовлении полупроводниковых устройств и осознанного научно-технического поиска.

Квантовые точки являются основой для формирования кубитов. Кубит (*q-бит*, кьюбит; от *quantum bit*) – квантовый разряд или наименьший элемент для хранения информации в квантовом компьютере. Как и бит, кубит допускает два собственных состояния, обозначаемых 0 и 1, но при этом может находиться и в их суперпозиции.

Упрощённая схема вычисления на квантовом компьютере выглядит так: берётся система кубитов, на которой записывается начальное состояние. Затем состояние системы или её подсистем изменяется посредством базовых квантовых операций. Для построения любого вычисления достаточно двух базовых операций. В конце измеряется значение, которое является результатом работы компьютера. Квантовая система даёт результат, являющийся правильным только с некоторой вероятностью. Но за счёт небольшого увеличения операций в алгоритме можно сколь угодно приблизить вероятность получения правильного результата к единице.

Теоретически квантовая схема вычислений может работать намного (в экспоненциальное число раз) быстрее классической. Большая часть современных компьютеров работает по такой схеме: n бит памяти хранят состояние, которое каждый такт времени изменяется процессором. В квантовом случае система из n кубитов находится в состоянии, являющемся суперпозицией всех базовых состояний, поэтому изменение системы касается всех $2n$ базовых состояний одновременно.

Благодаря огромной скорости разложения на простые множители, квантовый компьютер позволит расшифровывать сообщения, закодированные при помощи многих популярных криптографических алгоритмов. Так, сравнительно надёжным считается алгоритм разложения чисел на простые множители, так как эффективный способ такого разложения для классического компьютера в настоящее время неизвестен. Например, чтобы получить доступ к кредитной карте, нужно разложить на два простых множителя число длиной в сотни цифр. Даже для самых быстрых современных компьютеров выполнение этой задачи заняло бы больше времени, чем возраст Вселенной в сотни раз (более триллиарда лет). Эту задачу можно решить с использованием квантового компьютера. Методы квантовой криптографии открывают также новые возможности в области передачи секретных сообщений, которые даже теоретически нельзя расшифровать.

Квантовые точки, квантовые компьютеры и методы квантовой криптографии являются важными объектами для компьютерного моделирования и вычислительных экспериментов.

1.4. Программное обеспечение для математических расчетов

Существует несколько общеизвестных математических пакетов, реализующих различные численные методы, а также способных производить аналитические математические преобразования.

Mathcad (сайт разработчика – PTC – Parametric Technology Corporation, США, www.ptc.com/products/mathcad) является удобным инструментом работы для расчетчиков-инженеров, адаптирован к инженерным расчетам и имеет интуитивно понятный интерфейс. Пакет не предназначен для профессиональных математиков, которые используют системы, действующие в области символьной математики или математической статистики. В нем нет смысла программировать сложные задачи, для этого следует использовать систему Matlab или языки программирования Fortran или C++.

Владельцем программы MathCAD в настоящее является известный разработчик сквозных технологий проектирования CAD/CAE/CAM/PLM Pro/ENGINEER в машиностроении – компания PTC. В основе интеграции Mathcad и Pro/ENGINEER лежит двухсторонняя связь между этими приложениями. Их пользователи могут легко связать любой файл Mathcad с де-

талью и сборкой Pro/ENGINEER. В свою очередь, базовые величины, рассчитанные в системе Mathcad, могут быть переведены в параметры и размеры CAD-модели для управления геометрическим объектом. Параметры из модели Pro/ENGINEER также можно ввести в Mathcad для последующих инженерно-конструкторских расчётов. При изменении параметров взаимная интеграция двух систем позволяет динамически обновлять вычисления и конструкторскую документацию.

Отличительной особенностью программы MathCad является отсутствие специального языка программирования и связанная с этим легкость и наглядность решения задачи, возможность отображать математические формулы в том виде, в котором они обычно записываются на листе бумаги. Наличие развитых средств представления расчетных результатов упрощает анализ проекта и создание технических отчетов.

К недостаткам пакета следует отнести отсутствие встроенных средств отладки программ и невысокую скорость расчетов.

MATLAB (сокращённо от **Mat**rix **Lab**oratory – дословно в переводе «матричная лаборатория», сайт разработчика MathWorks, США, www.mathworks.com) ориентирован на решение научных и инженерных задач, построен по принципу языка программирования высокого уровня, позволяет сохранять документы в формате языка программирования С. MATLAB как высокоуровневая система программирования позволяет резко сократить затраты труда при проверке алгоритмов и проведении прикидочных расчётов. MATLAB работает как интерпретатор и включает большой набор инструкций (команд) для выполнения самых разнообразных вычислений, для задания структур данных и графического представления информации. Команды написаны на С, но много и таких команд, которые представлены в терминах встроенных С-программ. Пользователь может без особых затруднений добавлять свои команды и писать программы в терминах уже существующих команд; несколько сложнее делать это на Фортране и С. Можно обмениваться данными с программами на этих языках, а из них обращаться к системе. Имеются удобные способы управления счетом.

Главным недостатком программы является ее высокая стоимость для промышленного применения. Но доступны бесплатные студенческие лицензии. Разработчики программы расположены в Массачусетсе. Многие открытые курсы Массачусетского университета сопровождаются техниче-

скими расчетами в среде MATLAB и открывают свободный доступ к соответствующим файлам.

Mathematica (сайт разработчика Wolfram Research, США, www.wolfram.com) популярен среди теоретиков, предоставляет широкие возможности в проведении аналитических преобразований. Решение задачи осуществляется с помощью набора команд и по содержанию работы аналогично программированию. Существуют также приложения к программе во всех значимых инженерных направлениях.

Maple (сайт разработчика Maplesoft, Канада, www.maplesoft.com) решает задачи аналитически или численно. Программа способна обмениваться информацией с текстовыми редакторами для подготовки научных публикаций, активно позиционирует себя в мехатронике и конструировании машин.

Вопросы для самоконтроля

1. Напишите второй закон Ньютона в виде дифференциального уравнения. Какой физический смысл имеет коэффициент перед производной в этом уравнении?
2. Напишите все производные первого и второго порядка функции четырех переменных $f(t,x,y,z)$.
3. Составьте математическую модель для задачи о конструкционной прочности того кресла, на котором сидите. Включите в математическую модель дифференциальное уравнение, описывающее основное физическое явление, и граничные условия.

ТЕМА 2. Точность компьютерных вычислений

«Человеку свойственно делать ошибки,
но окончательно все запутать может только компьютер»
Правило точности Рэя (из законов Мерфи)

2.1. Источники погрешности расчетных результатов

Погрешность является мерой точности результата. Для количественной характеристики этой меры используют понятия абсолютной и относительной погрешностей. При работе с приближенными числами используют понятия значащих и верных значащих цифр. Значащая цифра приближенного числа называется верной, если абсолютная погрешность числа не пре-

восходит единицы разряда, соответствующего этой цифре. Абсолютную и относительную погрешности обычно записывают в виде числа, содержащего одну или две значащие цифры. При указании погрешностей округление всегда производится в большую сторону.

Существует четыре источника погрешности расчетных результатов: 1) ограничения физической и математической моделей процессов; 2) ошибочные или неполные исходные данные; 3) недостаточная точность используемых численных методов и 4) неточное округление промежуточных результатов при вычислениях.

1. *Погрешность модели* процессов связана с физическими допущениями и ограничениями, на базе которых формулируется математическая модель анализируемого инженерного объекта. Любая модель является лишь приближенным описанием реального процесса или явления. Погрешность, обусловленная неточностью модели, в ходе последующих вычислений не устраняется.

2. *Погрешность исходных данных* обусловлена погрешностью измерения технических параметров с помощью приборов. Приборная погрешность может достигать 10%. Такая погрешность является неустранимой и не может быть снижена математическими методами. Исходные данные для решения задачи, полученные экспериментально или в ходе предыдущих расчетов, часто являются основным источником погрешностей.

При вычислениях с приближенными числами важной задачей является оценка степени влияния погрешностей исходных данных на точность окончательного результата. Это необходимо не только для правильного учета вычислительных погрешностей, но также для определения возможных путей их уменьшения. Задачу вычисления величины y по известной величине x можно записать в виде

$$y = A(x) \quad , \tag{2.1}$$

где A – оператор, т.е. правило, согласно которому производится вычисление.

Ограничимся рассмотрением корректно поставленных задач вычисления. Задача вычисления $y = A(x)$ называется корректно поставленной, если решение задачи существует, является единственным и устойчивым по входным данным.

Остановимся на понятии устойчивости решения. Если при решении задачи возникает возмущение входных данных и вместо входной величины x оператор обрабатывает возмущенные входные данные с погрешностью $x + \delta x$, то мы получим в итоге возмущенное решение:

$$y + \delta y = A(x + \delta x) \quad . \tag{2.2}$$

Следовательно, погрешность входных данных порождает неустранимую погрешность решения:

$$\delta y = A(x + \delta x) - A(x) \quad . \tag{2.3}$$

Если решение непрерывно зависит от входных данных, то $|\delta y| \to 0$ всегда при выполнении условия $|\delta x| \to 0$, и тогда задача (2.1) устойчива по входным данным. Отсутствие устойчивости означает, что даже малым погрешностям δx могут соответствовать большие погрешности δy , а значит, результат вычислений может значительно отличаться от истинной величины. Применять непосредственно к такой неустойчивой задаче численные методы бессмысленно.

Не всякую формально устойчивую задачу следует безоговорочно подвергать численному решению. Пусть, например, выполняется условие $|\delta y| \le C|\delta x|$, где число C велико. Задача формально устойчива, но неустранимая ошибка решения может оказаться большой. Это случай плохой обусловленности задачи.

3. *Погрешность численных методов* появляется при переходе от аналитических зависимостей математической модели к дискретным функциям расчетной модели. Такая погрешность связана, например, с заменой интеграла суммой, производной – отношением разностей, функции – многочленом. Погрешность вычислительных методов является устранимой и должна быть в несколько раз меньше погрешности математического моделирования. На практике ограничиваются тем, чтобы довести погрешность используемых методов до величины, в несколько раз меньшей неустранимой погрешности. Дальнейшее повышение точности метода не приведет к повышению точности окончательного результата, а лишь увеличит трудоемкость расчетов из-за увеличения объема вычислений.

4. *Погрешность округления* возникает из-за ограниченной разрядности компьютеров. При решении с помощью компьютера сложных задач, в

которых вычисляется большое количество параметров для миллионов расчетных узлов, погрешность округления становится важным фактором. Такая погрешность накапливается в ходе вычислений и может достигать больших значений, когда выполняются миллиарды арифметических действий, и при этом алгоритм предполагает, например, вычитание близких по величине чисел. Погрешности округлений в сочетании с плохо организованным алгоритмом могут значительно исказить расчетные результаты.

2.2. Представление чисел в компьютерных вычислениях

Причиной появления вычислительных погрешностей является способ представления чисел в компьютере. Компьютерная арифметика отличается от традиционной математики дискретным представлением чисел. В традиционной математике числа могут иметь бесконечное количество цифр, например, в периодических числах. В компьютерной системе числа хранятся в регистрах и ячейках памяти с ограниченным количеством разрядов. Вследствие этого система вещественных чисел, участвующих в компьютерных расчетах, является дискретной и конечной. В компьютерной арифметике число представляется ограниченным количеством цифр.

Современные компьютеры позволяют обрабатывать целые и вещественные числа. По форме представления, способу хранения и реализации вычислительных операций в процессоре целые и вещественные числа существенно различаются.

При решении научных и инженерных задач в основном используются вещественные числа. Для отображения вещественных чисел, которые могут быть как очень маленькими, так и очень большими, используется форма записи чисел с порядком основания системы счисления. Любое вещественное число может быть представлено в степенном виде, который называется «стандартной формой с плавающей запятой»:

$$x = \pm b^c m, \tag{2.4}$$

где x – положительное или отрицательное вещественное число,

b – основание системы счисления (обычно используются основания 2, 8 или 16; основание всегда является положительной величиной: $b > 0$),

c – показатель степени, называется характеристикой или порядком,

m – мантисса.

Например, число с основанием 10 представляется в виде $x = \pm 10^c m$, с основанием 2 – в виде $x = \pm 2^c m$.

Для указания размерности двоичных данных используют общепринятые термины: бит – один разряд двоичного числа, который может принимать значение 0 или 1, байт – 8 битов.

Представление однобайтных двоичных целых чисел без знака (положительных) в прямом двоичном коде от 0 до 255 имеет вид:

$$x = \sum_{k=0}^{7} a_k 2^k , \qquad (2.5)$$

где a_i может принимать значение 0 или 1.

Распределение по разрядам коэффициентов суммы можно оформить в следующем виде:

a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7

Представление однобайтных двоичных целых чисел со знаком в прямом двоичном коде от – 127 … 0 … до + 127 требует выделения одного бита для отражения знака числа:

$$x = (-1)^S \sum_{k=0}^{6} a_k 2^k . \qquad (2.6)$$

S	a_0	a_1	a_2	a_3	a_4	a_5	a_6

Один бит (S) в представлении (2.6) содержит информацию о знаке числа: 0 – положительное, 1 – отрицательное; остальные биты служат для размещения модуля числа в прямом двоичном коде.

Приведенные примеры являются частными случаями представления чисел – только целых чисел – в двоичной системе счисления. В общем случае для вещественных чисел задают мантиссу и порядок числа.

Мантисса удовлетворяет условию нормировки:

$$b^{-1} \le m < b^0 \quad \text{или} \quad \frac{1}{b} \le m < 1, \quad \left[b = 2 \;\Rightarrow\; \frac{1}{2} \le m < 1 \right] , \qquad (2.7)$$

что обеспечивает единственность представления.

Мантисса определяет дробную часть числа и имеет вид конечной дроби по основанию b:

$$m = \frac{a_1}{b} + \frac{a_2}{b^2} + ... + \frac{a_k}{b^k}, \quad \left[b = 2 \ \Rightarrow \ m = \frac{a_1}{2} + \frac{a_2}{4} + ... + \frac{a_k}{2^k} \right], \qquad (2.8)$$

где k – точность представления мантиссы, задающая число отведенных в структуре разрядов; коэффициенты разложения мантиссы являются положительными числами: $0 \le a_i \le b-1, \ i = 1,...,k$.

Порядок задают в виде двоичного целого со знаком в прямом коде; мантиссу – в виде модуля нормализованного дробного числа в прямом коде с фиксированной точкой. Однобитовый знак мантиссы кодируют обычным образом – так же, как знак числа: 0 – положительное число, 1 – отрицательное.

Существует несколько международных стандартных форматов представления вещественных чисел, различающихся по точности, но имеющих одинаковую структуру, которая включает порядок (c_0, c_1) и знак порядка (P), знак мантиссы (S) и абсолютную величину мантиссы (a_1, a_2, a_3, a_4), например:

порядок			мантисса				
+ или -	2^0	2^1	+ или -	1/2	1/4	1/8	1/16
P	c_0	c_1	S	a_1	a_2	a_3	a_4

Коэффициенты c_0, c_1, a_1, a_2, a_3, a_4 принимают значение 0 или 1.

Чем больше разрядов отводится под запись мантиссы, тем выше точность представления числа. Чем больше разрядов занимает порядок, тем шире диапазон от наименьшего отличного от нуля числа до наибольшего числа, представимого в машине при заданном формате.

Возможны различные варианты структуры двоичных чисел, участвующих в компьютерных вычислениях. Так, в многобайтных числах последовательность байтов может выстраиваться слева направо (стандарт Motorola) или справа налево (стандарт Intel).

Пример 2.1. Представить вещественное число 8.75 в двоичной форме, т.е. с основанием $b = 2$. Описать порядок и мантиссу числа, используя 12 разрядов. Выделить под описание мантиссы 6 разрядов.

Решение. Чтобы определить порядок и мантиссу в двоичной системе счисления, делим число на 2 до получения остатка меньше 1. Таким образом, число можно представить в виде $8.75 = 0.546875 \cdot 2^4$, и его порядок равен 4, мантисса равна 0.546875. Дробная часть числа представляется в виде суммы $0.546875 = \frac{1}{2} + \frac{1}{32} + \frac{1}{64}$. Полученную информацию о 12-разрядном представлении числа 8.75 в двоичной системе счисления, можно свести в таблицу:

порядок						мантисса					
+ или -	2^0	2^1	2^2	2^3	2^4	+ или -	1/2	1/4	1/8	1/16	1/32
P	c_0	c_1	c_2	c_3	c_4	S	a_1	a_2	a_3	a_4	a_5
0	0	0	0	0	1	**0**	1	0	0	0	1

По условиям данного примера слагаемое $\frac{1}{64}$ в представлении мантиссы не может быть учтено в 12-разрядном представлении числа и образует ошибку – погрешность округления.

□

2.3. Ограничения машинных чисел

Входные данные помещаются в оперативную память компьютера в разрядном представлении. Некоторые вещественные числа, например иррациональные, невозможно записать в оперативную память точно, поскольку они имеют бесконечное количество цифр. Арифметические действия с участием таких чисел всегда производят ошибку вычисления. Те числа, которые могут быть помещены в оперативную память, называют машинными числами, их количество ограничено и зависит от количества разрядов, выделенных для записи числа.

Для описания машинных чисел и ограничений вводится 3 параметра, отвечающих за ошибку компьютерных вычислений и зависящих от разрядности:

1) минимальное машинное число ε_0,

2) относительная ошибка представления единицы ε_1,

3) максимальное машинное число ε_∞.

Минимальное машинное число (машинный нуль) выражается через основание и характеристику:

$$\varepsilon_0 = b^{c_{\min}-1}, \tag{2.9}$$

где $c_{\min}$ – минимальное возможное значение характеристики (отрицательное число).

Относительная ошибка представления единицы ε_1 выражается через основание и точность представления мантиссы:

$$\varepsilon_1 = b^{1-k} \quad . \tag{2.10}$$

Данный параметр отражает минимальное расстояние между двумя машинными числами на интервале от 1 до b; при вычислениях все числа вида $1+x$ из интервала $[1,\ 1+\varepsilon_1]$ заменяются машинным числом 1 с ошибкой, не превосходящей ε_1.

Максимальное машинное число (машинная бесконечность) равно

$$\varepsilon_\infty = b^{c_{\max}}(1-b^{-k}), \tag{2.11}$$

где $c_{\max}$ – максимальная возможная характеристика.

Из определения машинного числа можно сделать два вывода:

1) интервалы $(-\infty,\ -\varepsilon_\infty)$, $(-\varepsilon_0,\ 0)$, $(0,\ \varepsilon_0)$, $(1-\varepsilon_1/b,\ 1)$, $(1,\ 1+\varepsilon_1)$ и $(\varepsilon_\infty,\ \infty)$ не содержат машинных чисел;

2) точность представления машинных чисел можно повысить, увеличив количество цифр в мантиссе и расширив область допустимых значений характеристик.

В таблице приведены определяющие параметры для компьютеров с одинарной и двойной точностью представления машинных чисел для процессоров стандарта IEEE 754. Данный стандарт разработан ассоциацией IEEE (Institute of Electrical and Electronics Engineers) и используется для представления действительных чисел (чисел с плавающей точкой) в двоичном коде. Это наиболее используемый стандарт для вычислений с плавающей точкой, он используется многими микропроцессорами и логическими устройствами, а также программными средствами. Основное применение в технике и программировании получили форматы длиной 32 и 64 бита.

Параметр	*Одинарная точность*	*Двойная точность*
Основание системы счисления b	2	2
Минимальное значение характеристики $c_{\min}$	-125	-1021
Максимальное значение характеристики $c_{\max}$	128	1024
Точность представления мантиссы k	24	53
Минимальное машинное число ε_0	2^{-126}	2^{-1022}
Относительная ошибка представления единицы ε_1	2^{-23}	2^{-52}
Максимальное машинное число ε_∞	$2^{128}(1-2^{-24})$	$2^{1024}(1-2^{-53})$

Любое вещественное число $x \in (-\varepsilon_\infty, \varepsilon_\infty)$ в компьютере заменяется машинным числом x_m, для этого используется какой-нибудь способ приближения (усечение или округление). В результате такой замены возникает ошибка $x - x_m$, которая оценивается величиной $\varepsilon_1 |x|$ для вещественных чисел, модуль которых находится в интервале от наименьшего значения машинного числа до наибольшего $\varepsilon_0 \le |x| \le \varepsilon_\infty$. Ошибка зависит от минимального расстояния между числами: $|x - x_m| \le \varepsilon_1 |x|$. Числа меньше по модулю наименьшего машинного числа $|x| \in (0, \varepsilon_0)$ при замене машинным числом могут принимать одно из двух значений: 0 или ε_0. Тогда ошибка зависит от минимального машинного числа: $|x - x_m| \le \varepsilon_0 |x|$, которое называют также абсолютной ошибкой определения нуля. Эти две оценки можно объединить в выражение, которое справедливо для всех $x \in (-\varepsilon_\infty, \varepsilon_\infty)$:

$$x_m = x(1+\alpha) + \beta \ , \quad |\alpha| \le \varepsilon_1 \ , \quad |\beta| \le \varepsilon_0 \ . \tag{2.12}$$

Компьютер превращает исходные данные в машинные числа и затем выполняет арифметические действия над машинными числами. При вычислениях с плавающей точкой операция округления может потребоваться после выполнения любой из арифметических операций. Например, умножение или деление двух чисел сводится к умножению или делению мантисс. Так как в общем случае количество разрядов мантисс произведений и частных больше допустимой разрядности мантиссы, то требуется округление мантиссы результатов. При сложении или вычитании чисел с плавающей точкой эти числа должны быть предварительно приведены к одному

порядку, что осуществляется сдвигом вправо мантиссы числа, имеющего меньший порядок, и увеличением в соответствующее число раз порядка этого числа. Сдвиг мантиссы вправо может привести к потере младших разрядов мантиссы, таким образом, неизбежно появляется погрешность округления.

При вычислениях с помощью компьютера неизбежны погрешности округлений, связанные с ограниченностью разрядной сетки вычислительной машины. В некоторых случаях погрешности округлений в сочетании с плохо организованным алгоритмом могут сильно исказить результаты или даже привести к абсурдным результатам. В качестве примера рассмотрим решение простой системы двух уравнений с сохранением 6 цифр в расчетных результатах.

$$\begin{cases} -10^{-7}x + y = 1 \\ x + 5y = 6 \end{cases}$$

Решение можно начать с исключения переменной x из первого или из второго уравнения, в зависимости от этого решение окажется верным или неверным:

$$1)\ \begin{cases} x = 10^{7}y - 10^{7} \\ 10^{7}y - 10^{7} + 5y = 6 \end{cases} \Rightarrow \begin{cases} y = \dfrac{10^{7}+6}{10^{7}+5} \approx 1 \\ x \approx 0 \end{cases}$$; решение неверное;

$$2)\ \begin{cases} x = 6 - 5y \\ -10^{-7}(6-5y) + y = 1 \end{cases} \Rightarrow \begin{cases} y = \dfrac{1+6\cdot 10^{-7}}{1+5\cdot 10^{-7}} \approx 1 \\ x \approx 1 \end{cases}$$; решение верное.

Для компьютерных вычислений система уравнений представляется в матричном виде. В данном примере матричный вид уравнения включает квадратную матрицу коэффициентов и два вектора, в том числе вектор неизвестных параметров:

$$\begin{bmatrix} -10^{-7} & 1 \\ 1 & 5 \end{bmatrix} \begin{Bmatrix} x \\ y \end{Bmatrix} = \begin{Bmatrix} 1 \\ 6 \end{Bmatrix} .$$

В качестве резюме по теме. В настоящее время уже почти невозможно уследить за распространением компьютерных вычислений по прикладным областям. Такие вычисления проводятся в машиноведении и метрологии, химии и биологии, радиоэлектронике и электротехнике, геодезии

и геофизике, экономическом планировании и прогнозировании, теории автоматического управления и космических исследованиях, автоматизированном проектировании и машинной графике, разработке программного обеспечения общего назначения и даже в организации пожаротушения и огранке драгоценных камней.

Стремление к надежности численных расчетов привело не только к разработке нового математического аппарата и появлению многочисленных приложений, но и к развитию соответствующих компьютерных средств.

Вопросы для самоконтроля

1. Каковы основные источники погрешностей?
2. Что такое абсолютная и относительная погрешности?
3. Приведите к матричному виду систему уравнений:

$$\begin{cases} x + 2y + 3z = 4 \\ 5x + 6y + 7z = 8 \\ 9x + 10y + 11z = 12 \end{cases} .$$

ТЕМА 3. Численное интегрирование

«Человек, имеющий одни часы, твердо знает который час.
Человек, имеющий несколько часов, сомневается»
Из законов Мерфи

3.1. Концепция численного интегрирования

В случае аналитического решения многих задач, например, при вычислении момента инерции, вычисляют определенный интеграл. На выделенном интервале $[a,b]$ определенный интеграл I считается аналитически по формуле

$$I = \int_a^b f(x)dx \quad . \tag{3.1}$$

Численно такой определенный интеграл равен площади фигуры, ограниченной графиком функции $f(x)$ на интервале $[a,b]$, осью абсцисс и двумя нормалями к оси абсцисс $x = a$; $x = b$.

В компьютерном инжиниринге используется численное интегрирование. Численное интегрирование применяется к таблично заданным функциям, а также при взятии интегралов от достаточно сложных функций, которые предварительно приводятся к дискретному виду.

Все методы приближенного вычисления определенного интеграла основаны на том, что интервал интегрирования разбивается на подынтервалы. Затем на каждом подынтервале подынтегральная функция приближенно заменяется более простой функцией известного вида, например полиномом первого или второго порядка, т.е. такой функцией, от которой интеграл берется легко. При этом каждый участок исходного графика функции заменяется горизонтальной или наклонной прямой, параболой второго порядка или полиномиальной кривой более высокого порядка. На практике мы можем не знать аналитическую функцию, связывающую точки, поступившие на расчет; тогда точки функции, представленной дискретно, соединяются полиномами.

В результате получаются формулы интегрирования, называемые квадратурными, которые имеют вид взвешенной суммы ординат таблично заданной функции в отдельных точках:

$$I = \int_a^b f(x)dx \approx \sum_i \beta_i f(x_i) , \qquad (3.2)$$

где $f(x_i)$ – значения исходной функции в отдельных точках,

β_i – весовые коэффициенты соответствующих точек.

Чем меньше участки, на которых производят замену, тем точнее вычисляется интеграл. Поэтому исходный отрезок $[a,b]$ для повышения точности целесообразно разбить на множество равных (или неравных) участков, затем на каждом участке следует применить формулу интегрирования и сложить результаты.

На практике обычно требуется вычислить интеграл с некоторой заданной точностью, то есть должно выполняться условие $R \le \varepsilon$, где R – критерий точности, ε – допустимая величина погрешности вычислений.

Ошибка численного интегрирования зависит от количества участков разбиения, а значит – от величины шага. Погрешность численного интегрирования в большинстве случаев можно оценить, вычислив интеграл с исходным шагом $h = (b-a)/n$ для n интервалов разбиения области интегри-

рования $[a,b]$ и аналогичный интеграл с удвоенным шагом. Разница значений интеграла, вычисленного с исходным и уменьшенным вдвое количеством интервалов разбиения, определяет погрешность численного интегрирования.

Сравнение эффективности различных методов проводится по степени полинома, который может быть проинтегрирован данным методом точно, без ошибки. Чем выше степень интегрируемого без ошибки полинома, тем выше точность метода, тем эффективнее данный метод.

3.2. Метод прямоугольников

Метод прямоугольников является наиболее простым методом численного интегрирования. При использовании этого метода подынтегральная функция $f(x)$ на каждом участке разбиения заменяется линейной функцией, задающей на графике горизонтальную прямую (рис. 3.1) типа $y = c_0$, где c_0 – значение исходной функции в конечной точке элементарного участка слева (метод левых прямоугольников) или справа (метод правых прямоугольников).

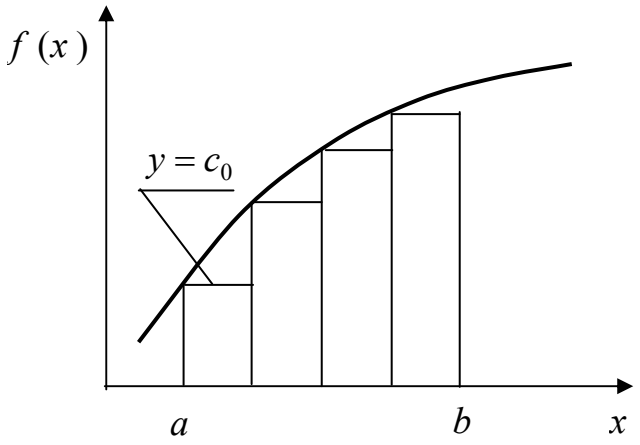


Рис. 3.1. Иллюстрация метода левых прямоугольников

При разбиении отрезка $[a,b]$ на n частей с равномерным шагом $h = (b-a)/n$ формула интегрирования методом левых прямоугольников имеет вид

$$I = \int_a^b f(x)dx \approx \sum_{i=0}^{n-1} f(x_i)h = h\sum_{i=0}^{n-1} f(x_i) , \qquad (3.3)$$

где шаг интегрирования представляет весовой коэффициент и является константой для всех точек исходной функции $h = \beta_i = const$.

Соответственно, метод правых прямоугольников можно выразить формулой

$$I = \int_a^b f(x)dx \approx \sum_{i=1}^{n} f(x_i)h = h\sum_{i=1}^{n} f(x_i) \ . \qquad (3.4)$$

Методом прямоугольников интеграл вычисляется абсолютно точно только для функций, представляемых горизонтальной линией: $f(x) = const$. Для других функций этот метод не находит большого практического применения в силу значительных погрешностей.

Пример 3.1. Вычислить интеграл $I = \int_0^1 x^3 dx$ методом левых прямоугольников и методом правых прямоугольников, разделив область интегрирования на 10 и на 100 участков. Оценить погрешность вычислений.

Решение. Аналитическое вычисление данного интеграла дает значение $I = \frac{1}{4}x^4\Big|_0^1 = 0.25$. Результаты дискретных расчетов сведены в таблицу.

Параметр	$n = 10$	$n = 100$
Интеграл, вычисленный методом левых прямоугольников	0.2025	0.245025
Интеграл, вычисленный методом правых прямоугольников	0.3025	0.255025
Интеграл, вычисленный методом левых прямоугольников с удвоенным шагом	0.1600	0.240100
Интеграл, вычисленный методом правых прямоугольников с удвоенным шагом	0.3600	0.260100
Оценка ошибки вычислений методом левых прямоугольников	0.05	0.005
Оценка ошибки вычислений методом правых прямоугольников	0.06	0.005

В качестве резюме по результатам расчетов. Как видно из таблицы, результат численного интегрирования при разбиении области определения на 10 элементарных участков является неудовлетворительным, а при разбиении на 100 участков обеспечивает точность во втором знаке.

□

При решении инженерных задач численными методами с использованием компьютерных программ всегда возникает, с одной стороны, проблема выбора между большим количеством расчетных точек, обеспечивающим точность решения, но требующим больших машинных ресурсов, и, с другой стороны, малым счетным временем компьютера, позволяющим находить решение быстро. Золотая середина обычно определяется «методом проб и оценки ошибок». Для выявления оптимального объема расчетной задачи проводится численное решение с небольшим количеством точек; затем количество точек увеличивается, решение повторяется, и так до достижения значения, при котором расчетный результат с учетом заданной точности перестает зависеть от количества расчетных точек.

3.3. Метод трапеций

В методе трапеций подынтегральная функция $f(x)$ заменяется линейной функцией, задающей на графике наклонную прямую (рис. 3.2) типа $y = c_1 x + c_0$.

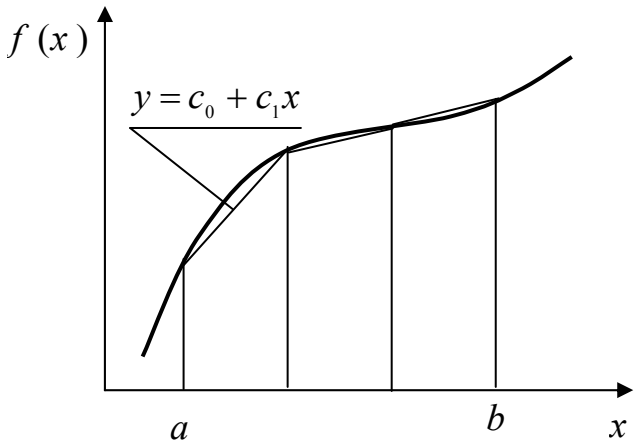


Рис. 3.2. Иллюстрация метода трапеций

Интеграл заменяется суммой площадей трапеций. Соответственно при разбиении отрезка $[a,b]$ на n частей с равномерным шагом $h = (b-a)/n$ формула интегрирования методом трапеций имеет вид

$$I = \int_a^b f(x)dx \approx \sum_{i=0}^{n-1} (f(x_i) + f(x_{i+1})) \frac{h}{2} = \\ = \frac{h}{2} f(x_0) + h \sum_{i=1}^{n-1} f(x_i) + \frac{h}{2} f(x_n) \quad , \tag{3.5}$$

где шаг интегрирования представляет весовой коэффициент и является константой для всех точек исходной функции $h = \beta_i = const$ за исключением крайних точек, для которых весовой коэффициент вдвое меньше: $\beta_0 = \beta_n = \frac{h}{2}$.

Методом трапеций можно точно вычислить только интегралы линейных и линейно-кусочных функций $f(x)$. По сравнению с методом прямоугольников метод трапеций является более точным, поскольку использует для описания исходных данных полином первого порядка.

Погрешность R вычислений интеграла методом трапеций можно оценить при использовании двойного просчета его значения с начальным и увеличенным вдвое шагом интегрирования:

$$R \le \frac{|I_{2n} - I_n|}{3}, \qquad (3.6)$$

где I_n и I_{2n} – соответственно приближенные значения интеграла при числе разбиений n и $2n$. Фактически при оценке погрешности R приближенное значение интеграла I_{2n} вычисляется с использованием всех значений таблично заданной функции $f(x)$, а при вычислении значения I_n каждое второе табличное значение функции в суммировании не участвует.

Можно начать приближенное вычисление интеграла с небольшим количеством шагов, и далее повторять вычисление, увеличивая число шагов. Удваивая на каждом этапе количество элементарных участков n и контролируя погрешность, можно подобрать такой шаг интегрирования, который обеспечивает заданную погрешность.

Пример 3.2. Вычислить интеграл $I = \int_0^1 x^3 dx$ методом трапеций. Найти шаг интегрирования, обеспечивающий точность вычислений интеграла в третьем знаке.

Решение. Аналитическое вычисление данного интеграла дает значение $I = \frac{1}{4} x^4 \Big|_0^1 = 0.25$.

Начнем приближенное вычисление интеграла, разбив отрезок [0,1] на 10 частей, $n = 10$. Затем последовательно станем удваивать число шагов,

каждый раз повторяя вычисление. Результаты произведенных расчетов сведены в таблицу.

Число разбиений n	Значение интеграла I	Ошибка вычислений R
10	0.062960	
20	0.250625	0.06
40	0.250156	0.0002
80	0.250039	0.00004

В качестве резюме по результатам расчетов. С увеличением числа разбиений расчетный результат стремится к точному значению, вычисленному аналитически. В произведенных расчетах точность вычисления данного интеграла в третьем знаке с использованием метода трапеций надежно достигается при разбиении области интегрирования на 40 элементарных участков.

□

3.4. Метод Симпсона

Метод базируется на замене подынтегральной функции квадратичным полиномом, задающим на графике параболу. Парабола строится на каждом элементарном участке по трем точкам (рис. 3.3) – по двум конечным точкам и одной срединной. По этим трем точкам определяют функцию параболы – полином второго порядка, который легко интегрируется аналитически и обеспечивает более высокую точность по сравнению с методом прямоугольников или трапеций. При решении задачи методом Симпсона интервал разбивают только на четное число участков.

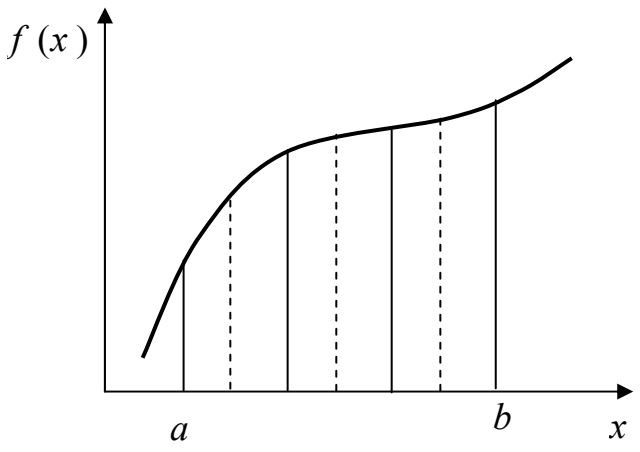


Рис. 3.3. Иллюстрация метода Симпсона

При разбиении отрезка $[a,b]$ на $2n$ частей с равномерным шагом $h=(b-a)/2n$ формула интегрирования методом Симпсона имеет вид

$$I=\int_a^b f(x)dx\approx$$

$$\approx\frac{hf(x_0)}{3}+\frac{4h}{3}\sum_{i=1}^{n}f(x_{2i-1})+\frac{2h}{3}\sum_{i=1}^{n-1}f(x_{2i})+\frac{hf(x_{2n})}{3}= \qquad (3.7)$$

$$=\sum_{i=0}^{2n}\beta_i f(x_i) \quad ,$$

где шаг интегрирования обусловливает весовой коэффициент, который различается для ординат с нечетными и четными номерами: $\beta_0=\beta_n=\frac{h}{3}$,

$\beta_1=\beta_3=...=\beta_{n-1}=\frac{4h}{3}$, $\beta_2=\beta_4=...=\beta_{n-2}=\frac{2h}{3}$.

Погрешность R вычислений интеграла методом Симпсона можно оценить при выполнении двойного просчета его приближенного значения с начальным и увеличенным вдвое шагом интегрирования:

$$R\le\frac{|I_{2n}-I_n|}{15}\ , \qquad (3.8)$$

где I_{2n} и I_n – соответственно значения интеграла при количестве участков разбиения $2n$ и n. Удваивая на каждом этапе количество элементарных участков и контролируя погрешность, можно подобрать такой шаг интегрирования, который обеспечивает заданную погрешность.

Пример 3.3. Вычислить приближенно интеграл $I=\int_0^1\frac{dx}{1+x^2}$ методом Симпсона. Найти шаг интегрирования, обеспечивающий точность вычислений интеграла в третьем знаке.

Решение. Аналитическое вычисление данного интеграла дает значение $I=arctg(x)\Big|_0^1=0.7853981634$. Начнем численное определение интеграла с разбиения отрезка [0,1] на 2 элементарных участка, $n=2$; тогда $x_0=0,\ x_1=0.5,\ x_2=1,\ h=0.5$. Затем будем последовательно удваивать

число шагов интегрирования и повторять вычисления. Результаты произведенных расчетов сведены в таблицу.

Число разбиений n	Значение интеграла I	Ошибка вычислений R
2	0.783333333	
4	0.785392157	0.002
8	0.785398126	0.000006

С увеличением числа разбиений расчетный результат стремится к точному значению, вычисленному аналитически.

□

Точность приближенного вычисления интегралов с использованием метода Симпсона достигается при значительно меньшем числе разбиений области интегрирования по сравнению с методом прямоугольников или методом трапеций.

3.5. Метод Ньютона-Котеса

Метод предполагает замену подынтегральной функции полиномом k-го порядка. Расчетная формула для одного элементарного участка имеет вид

$$I = \int_a^b f(x)dx \approx (b-a)\sum_{i=0}^{k} f(x_i)H_i = \sum_{i=0}^{k} \beta_i f(x_i) , \qquad (3.9)$$

где $k+1$ – число ординат, используемых для аппроксимации подынтегральной функции; и весовые коэффициенты β_i слагаемых зависят от коэффициентов Ньютона-Котеса H_i : $\beta_i = (b-a)H_i$, $x_0 = a$, , $x_i = a + i(b-a)/k$.

Для замены подынтегральной функции полиномом третьей степени потребуется 4 точки, полиномом четвертой степени – 5 точек и т.д. Коэффициенты Ньютона-Котеса H_i не зависят от функции $f(x)$, определяются только порядком полинома k; и это же значение равно необходимому количеству интервалов разбиения. Коэффициенты Ньютона-Котеса приведены в таблице.

Порядок полинома k	Коэффициенты Ньютона-Котеса H_i
1	$H_0 = H_1 = \frac{1}{2}$
2	$H_0 = H_2 = \frac{1}{6} \quad H_1 = \frac{2}{3}$
3	$H_0 = H_3 = \frac{1}{8} \quad H_1 = H_2 = \frac{3}{8}$
4	$H_0 = H_4 = \frac{7}{90} \quad H_1 = H_3 = \frac{16}{45} \quad H_2 = \frac{2}{15}$
5	$H_0 = H_5 = \frac{19}{288} \quad H_1 = H_4 = \frac{25}{96} \quad H_2 = H_3 = \frac{25}{144}$

Можно убедиться в том, что методы численного интегрирования, рассмотренные выше, являются частным случаем метода Ньютона-Котеса, а именно: при $k = 1$ получаем метод трапеций, при $k = 2$ получаем метод Симпсона. Метод Ньютона-Котеса дает устойчивый результат при $k < 10$. Большие значения порядка брать не следует.

При разбиении всего интервала на n элементарных участков формулу (3.9) можно применить для каждого участка, а полученные результаты вычислений сложить.

3.6. Метод Чебышева

Ранее мы рассмотрели формулы интегрирования с постоянным шагом, они имеют простой вид, но их использование не всегда приемлемо с точки зрения вычислительной эффективности. Метод Чебышева предполагает одинаковое значение весового коэффициента для всех шагов интегрирования, но при этом интервал интегрирования $[a,b]$ разбивается на элементарные участки произвольно. Сама переменная интегрирования трансформируется $x \to z$ и приводится к диапазону $[-1,1]$ согласно следующему правилу:

$$x = \frac{a+b}{2} + \frac{b-a}{2} z \quad . \tag{3.10}$$

Формула численного интегрирования методом Чебышева имеет вид

$$I = \int_a^b f(x)dx = \int_{-1}^{1} f(z)dz \approx \frac{b-a}{k}\sum_{i=1}^{k} f(z_i), \qquad (3.11)$$

где коэффициенты k обозначают число ординат, использующихся при расчетах на отдельных участках. Полином можно интегрировать без ошибки при значениях k от 2 до 9. Соответствующие коэффициентам k параметры z_i приведены в таблице.

Число ординат k	Приведенные абсциссы z_i
2	$-z_1 = z_2 = 0.577350$
3	$-z_1 = z_3 = 0.707107 \quad z_2 = 0$
4	$-z_1 = z_4 = 0.794654 \quad -z_2 = z_3 = 0.187592$
5	$-z_1 = z_5 = 0.832498 \quad -z_2 = z_4 = 0.374541 \quad z_3 = 0$

3.7. Метод Гаусса

Метод не фиксирует в качестве констант ни весовые коэффициенты, ни шаги разбиения интервала интегрирования. Переменная интегрирования трансформируется $x \to z$ и приводится к диапазону $[-1,1]$ по формуле (3.10) аналогично методу Чебышева. Все весовые коэффициенты β_i и приведенные абсциссы z_i в методе Гаусса могут иметь разные значения. Формула численного интегрирования методом Гаусса имеет вид

$$I = \int_a^b f(x)dx = \int_{-1}^{1} f(z)dz \approx \frac{b-a}{k}\sum_{i=1}^{k} \beta_i f(z_i), \qquad (3.12)$$

где соответствующие коэффициентам k параметры z_i и β_i приведены в таблице.

Число ординат k	Приведенные абсциссы z_i	Весовые коэффициенты β_i
2	$-z_1 = z_2 = 0.577350$	$\beta_1 = \beta_2 = 1$
3	$-z_1 = z_3 = 0.774597 \quad z_2 = 0$	$\beta_1 = \beta_3 = 0.555555 \quad \beta_2 = 0.888889$
4	$-z_1 = z_4 = 0.861136$ $-z_2 = z_3 = 0.339981$	$\beta_1 = \beta_4 = 0.347855$ $\beta_2 = \beta_3 = 0.652145$

Вопросы для самоконтроля

1. Можно ли методом прямоугольников получить точное значение определенного интеграла?
2. Как уменьшить в методе трапеций погрешность нахождения определенного интеграла?
3. Найдите приближенное значение интеграла $I = \int_0^1 x^3 dx$, разбив участок интегрирования на 10 частей, методом левых прямоугольников, методом трапеций, методом Симпсона. Оцените погрешность численного интегрирования.

ТЕМА 4. Интерполяция

«В любом наборе исходных данных самая надежная величина,
не требующая никакой проверки,
является ошибочной»
Из законов Мерфи

4.1. Концепция интерполяции

Дискретное решение инженерных задач с использованием компьютерных программ требует введения исходных данных для расчета в виде таблично заданных функций. Значения дискретной функции получают также при измерении различных параметров и свойств с помощью приборов. Введенные или полученные в расчете дискретные данные при их дальнейшей обработке или при выполнении вычислений с их участием требуют, как правило, интерполяции. Основная задача интерполяции заключается в замене таблично заданной функции простой аналитической функцией и затем нахождение с ее помощью приближенных значений в тех точках внутри интервала, где исходная функция не задана.

Существующие данные можно использовать также для того, чтобы приближенно оценить возможные значения параметра за пределами определенной области. Такую «экспансию» за пределы исходной области определения осуществляют с помощью экстраполяции. Под экстраполяцией понимают приближенное восстановление функции в точках за пределами заданного интервала.

Важное применение интерполяция и экстраполяция находят в обработке экспериментальных данных, полученных, например, исследователями в измерениях или инженерами при контролировании технологических процессов. Графическая интерполяция линиями Безье используется во всех компьютерных программах, ориентированных на конструирование и создание изображений методами векторной графики.

Решение задачи интерполяции обеспечивается построением аналитической интерполяционной функции $L(x)$, приближенно заменяющей исходную функцию $f(x_i)$, заданную таблично. Интерполяционная функция проходит через все заданные исходные точки (рис. 4.1). Исходные точки называют *узлами интерполяции*.

С помощью интерполяционной функции можно распространить таблично заданную функцию на те области, в которых точное значение исходной функции неизвестно, и таким образом приближенно вычислить величину искомого параметра в произвольной точке, не совпадающей с интерполяционным узлом.

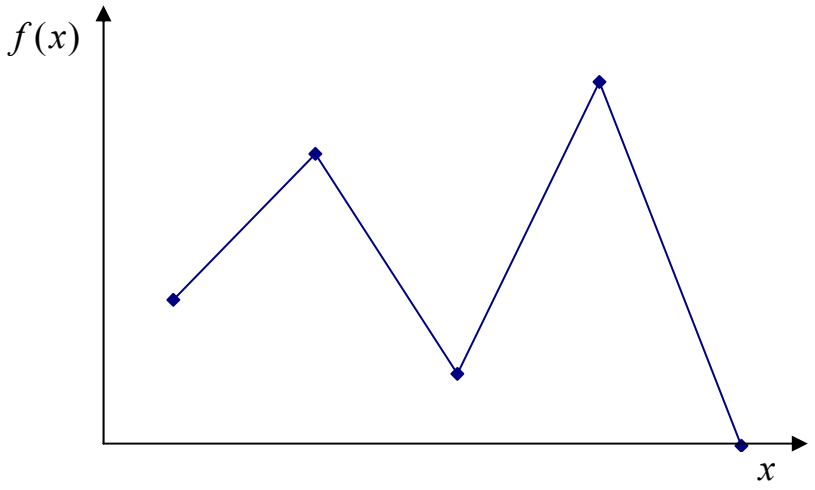


Рис. 4.1. Интерполяция дискретной функции
линейными интерполяционными функциями

В связи с выполнением интерполяционной процедуры формулируются и решаются следующие задачи:

1) выбор наилучшей интерполяционной функции $L(x)$;

2) оценка погрешности интерполяции $R(x)$;

3) оптимальное размещение узлов интерполяции $(x_0, x_1, ..., x_n)$ для обеспечения наибольшей возможной точности восстановления таблично заданной функции.

Первая из перечисленных проблем интерполяции чаще всего решается выбором в качестве интерполяционной функции полинома $L_n(x)$, проходящего через заданные точки. С вычислительной точки зрения полиномы имеют полезные свойства, а именно, производные и интегралы от полиномов также являются полиномами. Это означает, что вычисления с использованием полиномов в каждой точке сводятся к выполнению только операций умножения и сложения.

Все интерполяционные методы, базирующиеся на использовании полиномов в качестве интерполяционной функции, приводят в итоге к одному единственно возможному результату. Это объясняется тем, что полином n-й степени, содержащий $n+1$ параметр и проходящий через все заданные $n+1$ точки, – единственный. Более того, полином можно представить как усеченный ряд Тейлора, в который разложили исходную дифференцируемую функцию. Существование и единственность решения является главным достоинством интерполяции полиномами.

В общем виде полином степени n имеет вид

$$P_n(x) = c_0 + c_1 x + ... + c_{n-1} x^{n-1} + c_n x^n = \sum_{k=0}^{n} c_k x^k \quad . \qquad (4.1)$$

В качестве интерполяционных функций могут быть использованы также тригонометрические, экспоненциальные или другие функции. Выбор той или иной интерполяционной функции обусловлен конкретной постановкой задачи и ожидаемым изменением анализируемого параметра. Изменение параметра определяется его физической природой. Во многих случаях физическая природа искомого параметра известна. Например, температура изменяется по экспоненте, а упругая деформация нарастает линейно.

4.2. Метод Лагранжа

Метод Лагранжа использует в качестве интерполяционной функции полином n-го порядка и позволяет таблично заданную функцию $f(x_i) = y_i$, $i = 0, ..., n$, значения которой известны в $n+1$ точках, восстановить в произвольной точке x, принадлежащей отрезку $[x_0, x_n]$.

Полином Лагранжа n-го порядка имеет вид

$$L_n(x) = f(x_0)A_0(x) + ... + f(x_i)A_i(x) + ... + f(x_n)A_n(x) = \sum_{i=0}^{n} y_i A_i(x), \quad (4.2)$$

где коэффициенты полинома $A_i(x) = \dfrac{(x-x_0)...(x-x_{i-1})(x-x_{i+1})...(x-x_n)}{(x_i-x_0)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)}$ содержат в числителе n сомножителей первого порядка, а в знаменателе – произведение чисел; причем в перемножаемых разностях числителя и знаменателя присутствуют все интерполяционные узлы за исключением текущего.

Нетрудно заметить, что при вычислении полинома непосредственно в интерполяционных узлах, когда заданный аргумент интерполяционной функции равен $x = x_j$, коэффициенты полинома принимают значения:

$$A_i(x_j) = \frac{(x_j-x_0)...(x_j-x_j)...(x_j-x_{i-1})(x_j-x_{i+1})...(x_j-x_n)}{(x_i-x_0)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)} = 0, \text{ если } i \neq j;$$

поскольку в этом случае в числителе присутствует разность $(x_j - x_j)$, и

$$A_i(x_j) = A_i(x_i) = \frac{(x_i-x_0)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)}{(x_i-x_0)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)} = 1, \text{ если } i = j;$$

поскольку в этом случае числитель равен знаменателю.

Тогда сам полином в интерполяционных узлах принимает соответствующие значения исходной функции: $L_n(x_j) = f(x_j)$.

Пример 4.1. Определить полином Лагранжа для линейной функции, проходящий через две точки (x_0, y_0) и (x_1, y_1).

Решение. Линейная функция интерполируется полиномом первого порядка:

$$L_1(x) = y_0 A_0(x) + y_1 A_1(x) = y_0 \frac{x-x_1}{x_0-x_1} + y_1 \frac{x-x_0}{x_1-x_0} \quad , \quad (4.3)$$

после преобразования получим линейную интерполяционную функцию, которая принимает значения исходной таблично заданной функции в интерполяционных узлах, причем полиномиальные коэффициенты выражаются через координаты интерполяционных узлов:

$$L_1(x) = \frac{y_1-y_0}{x_1-x_0}x + \frac{y_0x_1 - y_1x_0}{x_1-x_0}. \quad (4.4)$$

□

Решая пример 4.1, следует обратить внимание на очевидный способ построения интерполяционной функции: из условия прохождения функции через все точки можно составить систему уравнений типа $y_j = \sum_i a_i x_j^i$, из решения которой непосредственно вычисляются полиномиальные коэффициенты. При малом количестве интерполяционных узлов такой метод приемлем, но он становится неэффективным в случае большого числа анализируемых точек. Когда количество интерполяционных узлов велико, к явному вычислению коэффициентов полинома не прибегают, вместо этого используют вычислительную процедуру, основанную на определении вспомогательных полиномов согласно рекуррентному соотношению.

Пример 4.2. Горизонтальная балка длиной 1 м с квадратным сечением и высотой 0.01 м закреплена в точке $x = 0$; в точке $x = 1$ м приложена сила 100 Н, направленная вертикально вверх, противоположно ускорению гравитации. Известны вертикальные перемещения y_i рассматриваемой балки в нескольких точках x_i; они приведены в таблице.

Провести интерполяцию методом Лагранжа известных перемещений и вычислить вертикальное перемещение балки в точке с координатой $x = 0.9$ м с использованием 2, 3 и 4 интерполяционных узлов.

Количество узлов интерполяции								
$N = 2$			$N = 3$			$N = 4$		
i	x_i, м	y_i, см	i	x_i, м	y_i, см	i	x_i, м	y_i, см
0	1.0	0.20080	0	1.0	0.20080	0	1.0	0.20080
1	0.8	0.14137	1	0.8	0.14137	1	0.8	0.14137
			2	0.7	0.11315	2	0.7	0.11315
						3	0.6	0.08675

Решение. Распишем полином Лагранжа и вычислим значение интерполяционной функции в искомой точке. Если число узлов интерполяции равно N, то порядок соответствующего интерполяционного полинома n будет на единицу меньше: $n = N - 1$.

$N = 2$

$$L_1(x) = y_0 A_0(x) + y_1 A_1(x) = y_0 \frac{x - x_1}{x_0 - x_1} + y_1 \frac{x - x_0}{x_1 - x_0}$$

$$L_1(x = 0.9) = 0.17109$$

$$N = 3$$

$$L_2(x) = y_0 A_0(x) + y_1 A_1(x) + y_2 A_2(x) =$$

$$= y_0 \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + y_1 \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + y_2 \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

$$L_2(x = 0.9) = 0.17059$$

$$N = 4$$

$$L_3(x) = y_0 A_0(x) + y_1 A_1(x) + y_2 A_2(x) + y_3 A_3(x) =$$

$$= y_0 \frac{(x - x_1)(x - x_2)(x - x_3)}{(x_0 - x_1)(x_0 - x_2)(x_0 - x_3)} + y_1 \frac{(x - x_0)(x - x_2)(x - x_3)}{(x_1 - x_0)(x_1 - x_2)(x_1 - x_3)} +$$

$$+ y_2 \frac{(x - x_0)(x - x_1)(x - x_3)}{(x_2 - x_0)(x_2 - x_1)(x_2 - x_3)} + y_3 \frac{(x - x_0)(x - x_1)(x - x_2)}{(x_3 - x_0)(x_3 - x_1)(x_3 - x_2)}$$

$$L_3(x = 0.9) = 0.17079$$

Результаты вычислений сведены в таблицу. Известно экспериментальное перемещение балки в данной точке: $y(x = 0.9) = 0.17078$, поэтому есть возможность оценить точность расчетов. В таблице сведены расчетные значения перемещений и приведены значения ошибки приближения для рассмотренных вариантов интерполяционных полиномов. Наиболее близким к известному решению является результат интерполяции по четырем точкам.

Количество узлов интерполяции N	Вычисленное вертикальное перемещение, см y	Абсолютная ошибка вычислений, см Δy
2	0.17109	0.00031
3	0.17059	-0.00020
4	0.17079	0.00001

□

Интерполяционные полиномы Лагранжа имеют простой вид и часто используются для теоретического анализа. Однако с точки зрения практи-

ческих вычислений рассмотренное здесь нерекуррентное представление применимо только для малого количества интерполяционных узлов n, как правило, не больше 10.

С увеличением n резко увеличивается объем вычислений, при этом сомножители в полиноме становятся громоздкими и начинают сильно осциллировать из-за накопления арифметических ошибок.

4.3. Метод Ньютона

Интерполяция по формулам Ньютона эффективна при обработке большого объема табличных данных и равномерном расположении интерполяционных узлов. Расстояние между узлами должно быть одинаковым. Метод позволяет прогнозировать значение дискретной функции, не размышляя о выборе и построении интерполяционной функции. В методе используется понятие конечных разностей, которые для дискретных функций являются аналогами дифференциалов непрерывных функций. Пусть исходная функция задана таблично для $n+1$ точек $f(x_i) = y_i$, $i = 0,...,n$, расположенных равномерно на числовой оси согласно условию $x_{i+1} = x_i + h$, причем шаг по оси абсцисс можно выразить через крайние значения интервала: $h = (x_n - x_0)/n$.

Определение конечных разностей. Конечные разности нулевого порядка $\Delta^0 y_i$ равны значениям функции в интерполяционных узлах: $\Delta^0 y_i = y_i$. Конечные разности первого порядка равны разностям между значениями функции в соседних узлах интерполяции $\Delta^1 y_i = y_{i+1} - y_i$.

Используя конечные разности первого порядка, можно вычислить конечные разности второго порядка:

$$\Delta^2 y_i = \Delta^1 y_{i+1} - \Delta^1 y_i = (y_{i+2} - y_{i+1}) - (y_{i+1} - y_i) \ . \qquad (4.5)$$

По аналогии можно продолжить вычисление конечных разностей более высоких порядков. Методом математической индукции можно доказать, что конечная разность k-го порядка в i-й точке вычисляется через конечные разности более низкого порядка $(k-1)$:

$$\Delta^k y_i = \Delta^{k-1} y_{i+1} - \Delta^{k-1} y_i \ . \qquad (4.6)$$

Ограничение конечных разностей. При наличии $n+1$ точек в исходных данных конечную разность первого порядка можно вычислить для первых n точек; а конечную разность k-го порядка – только для первых $(n-k+1)$ точек.

С учетом конечных разностей интерполяционный многочлен Ньютона представляется следующим выражением:

$$P_n(x) = y_0 + \frac{\Delta y_0 (x - x_0)}{h} + \frac{\Delta^2 y_0 (x - x_0)(x - x_1)}{2! h^2} + \\ + \frac{\Delta^3 y_0 (x - x_0)(x - x_1)(x - x_2)}{3! h^3} + \ldots + \frac{\Delta^n y_0 (x - x_0)(x - x_1)\ldots(x - x_{n-1})}{n! h^n} \quad . \tag{4.7}$$

Приведенная формула носит название первой интерполяционной формулы Ньютона и рекомендуется для применения при интерполяции вперед, т.е. в сторону увеличения x, или при экстраполяции назад, т.е. левее x_0 на числовой оси. Конечные разности в формуле Ньютона представляют собой числовые коэффициенты, рассчитанные по исходным значениям таблично заданной функции и по величине шага интерполяции, который, как мы помним, задается постоянным.

Формула Ньютона может быть записана через безразмерную переменную $q = \frac{x - x_0}{h}$, показывающую, сколько содержится шагов от начальной точки x_0 до заданной точки x:

$$P_n(x) = y_0 + \Delta y_0 q + \frac{\Delta^2 y_0 q(q-1)}{2!} + \\ + \frac{\Delta^3 y_0 q(q-1)(q-2)}{3!} + \ldots + \frac{\Delta^n y_0 q(q-1)\ldots(q-n+1)}{n!} \quad . \tag{4.8}$$

Следует отметить, что в формуле Ньютона (4.7) безразличен порядок нумерации узлов, и это дает возможность подключать к исходным данным дополнительные узлы с постоянным заданным шагом h для построения полинома более высокого порядка.

При вычислениях по формуле Ньютона (4.8) слагаемые ряда добавляются в сумму последовательно с увеличением порядка вычисляемых конечных разностей; при этом точность расчетов удобно оценивать, наблюдая за тем, насколько быстро убывают члены ряда. Если это происходит

достаточно быстро, в сумме можно оставлять только те слагаемые, которые больше заданной погрешности расчетов.

Погрешность интерполяции методом Ньютона можно оценить по формуле

$$R(x) = \left| f(x) - P_n(x) \right| \le \frac{M_{n+1}}{(n+1)!} \prod_{i=0}^{n} (x - x_i) =$$

$$= \frac{M_{n+1} h^{n+1}}{(n+1)!} q(q-1)(q-2)...(q-n) = \tag{4.9}$$

$$= \frac{\Delta^{n+1} y_0}{(n+1)!} q(q-1)(q-2)...(q-n),$$

где $M_{n+1} = \max \left| f^{(n+1)}(x) \right|$ – максимальное значение $(n+1)$-й производной исходной функции $f(x)$ на отрезке между наименьшим и наибольшим из значений аргумента $[x_0, x_n]$.

Согласно формуле (4.8) с уменьшением расстояния между узлами интерполяции (шага) h погрешность представления функции полиномом Ньютона убывает. Для оценки этой погрешности необходима некоторая дополнительная информация об исходной функции; такой информацией является производная $(n+1)$-го порядка, для вычисления которой потребуется несколько дополнительных точек x_{n+1}, x_{n+2}, … . Производные таблично заданной функции с постоянным шагом h можно выразить через конечные разности Δy и расстояние между интерполяционными узлами h следующим образом:

$$\frac{df}{dx} \approx \frac{\Delta y}{h}, \quad \frac{d^2 f}{dx^2} \approx \frac{\Delta^2 y}{h^2}, \quad \dots \quad \frac{d^k f}{dx^k} \approx \frac{\Delta^k y}{h^k} . \tag{4.10}$$

Пример 4.3. Горизонтальная балка длиной 1 м с квадратным сечением и высотой 0.01 м закреплена в точке $x = 1$ м, в точке $x = 0$ приложена сила 100 Н, направленная вертикально вверх (противоположно ускорению гравитации).

Известны вертикальные перемещения y_i рассматриваемой балки в нескольких точках x_i; они приведены в таблице.

Провести интерполяцию методом Ньютона и вычислить вертикальное перемещение в точке с координатой $x = 0.05$ м; оценить погрешность вычислений.

x, м	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7
y, см	0.20080	0.17078	0.14137	0.11315	0.08675	0.06275	0.04177	0.02440

Решение. Используем для интерполяции четыре первые точки, остальные точки используем для оценки погрешности.

Порядок интерполяционного полинома $n = 3$. Шаг аргумента исходной таблично заданной функции $h = 0.1$ м. Значение относительного аргумента, определяющего количество шагов до исследуемой точки с координатой $x = 0.05$ м, равно $q = (x - x_0)/h = 0.5$.

Составим таблицу конечных разностей, заполняя текущий столбец вычитанием ячеек в предыдущем столбце исходя из того правила, что конечная разность нулевого порядка равна значению функции. Далее конечная разность первого порядка вычисляется как разность значений функции; а конечная разность второго порядка получается вычитанием конечной разности первого порядка текущей точки из конечной разности первого порядка последующей точки. По аналогии вычисляются конечные разности более высокого порядка. Результаты вычислений приведены в таблице.

x, м	y, см	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$
0	0.20080	-0.03002	0.00061	0.00058	0.00005
0.1	0.17078	-0.02941	0.00119	0.00063	-0.00005
0.2	0.14137	-0.02822	0.00182	0.00058	0.00004
0.3	0.11315	-0.02640	0.00240	0.00062	-0.00003
0.4	0.08675	-0.02400	0.00302	0.00059	
0.5	0.06275	-0.02098	0.00361		
0.6	0.04177	-0.01737			
0.7	0.02440				

Произведем вычисление по формуле Ньютона:

$$P_n(x=0.05)=P_n(q=0.5)=$$
$$=y_0+\Delta y_0 q+\frac{\Delta^2 y_0 q(q-1)}{2!}+\frac{\Delta^3 y_0 q(q-1)(q-2)}{3!}=$$
$$=0.20080+(-0.03002)\cdot 0.5+0.00061\cdot 0.5\cdot(1-0.5)/2!+$$
$$+0.00058\cdot 0.5\cdot(1-0.5)\cdot(2-0.5)/3!=0.18575 \quad .$$

Оценим погрешность найденного значения по максимальному модулю конечной разности четвертого порядка. Из таблицы находим, что конечная разность четвертого порядка $\Delta^4 y_0 = 0.00005$, тогда погрешность интерполяции равна

$$R(x=0.05)=\left|f(x)-P_n(x)\right|\le\frac{M_{n+1}h^{n+1}}{(n+1)!}q(q-1)(q-2)...(q-n)=$$
$$=\frac{\Delta^4 y_0}{4!}q(q-1)(q-2)(q-3)(q-4)=0.000007 \quad .$$

Необходимо обратить внимание на тот факт, что конечные разности, и соответственно производные исходной функции, начиная с третьего порядка, осциллируют. Следовательно, надежной в данном случае можно признать интерполяцию полиномами Ньютона не выше второго порядка. Вычисленные значения вертикальных перемещений балки в исследуемой точке $x=0.05$ и соответствующие погрешности вычислений при использовании полиномов разного порядка приведены в таблице.

Число узлов интерполяции N	Порядок полинома n	Вычисленное значение, см y	Оценка ошибки R
2	1	0.18575	0.000007
3	2	0.18571	0.0001
4	3	0.18579	0.0007

Значения вертикального перемещения балки в заданной точке, вычисленные интерполяцией по полиномам первого, второго и третьего порядков, различаются в пятом знаке после запятой, что удовлетворяет точности в определении вертикального прогиба (в см) для стальной балки длиной 1 м и высотой 1 см. По самой грубой оценке вертикальное перемещение балки в заданной точке составляет 1.86 мм с погрешностью 0.01 мм.

Точность проведенных вычислений, исходя из физического смысла определяемого параметра, является достаточной.

□

4.4. Интерполяция сплайнами

При большом количестве узлов интерполяции приходится использовать интерполяционные полиномы высокой степени, что создает проблемы при организации вычислительных операций. Высокой степени интерполяционного полинома можно избежать, если разбить весь интервал определения на несколько подынтервалов и на каждом подынтервале применить свою интерполяционную функцию. Такой «кусочный» подход требует «сшивки» интерполяционных функций для соседних подынтервалов в точках перехода, чтобы обеспечить непрерывность суммарной интерполяционной функции и ее первой производной. Наиболее распространенным методом «кусочной» интерполяции является сплайн-интерполяция.

Слово «сплайн» переводится как «гибкая линия». Сплайн можно использовать для проведения кривой линии через заданную совокупность точек, изгибая линию между точками и добиваясь, чтобы она проходила через все рассматриваемые точки. Строго говоря, сплайн – это функция, которая на каждом межузловом подынтервале совпадает с некоторым полиномом, своим для каждого подынтервала. Полиномы соседних подынтервалов стыкуются в граничных точках таким образом, чтобы интерполяционная функция в целом была непрерывной.

Наибольшее распространение получила интерполяция с помощью кубических сплайнов. Кубический сплайн «склеивается» из полиномов третьей степени, которые для любого i-го участка записываются в виде

$$y = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \ . \qquad (4.11)$$

Соответственно на всем интервале определения исходной функции будет построено n кубических полиномов, отличающихся коэффициентами a_i, b_i, c_i, d_i. Можно доказать, что задача нахождения кубического сплайна имеет единственное решение.

Рассмотрим вариант сплайновой интерполяции в случае равномерного расположения интерполяционных узлов, т.е. для исходных данных выполняется условие: $x_{i+1} - x_i = const = h$. Чтобы вычислить четыре полино-

миальных коэффициента на текущем участке, следует составить четыре уравнения, которые отражают следующие условия:

1) условие прохождения сплайна через начальную точку (x_i, y_i) каждого участка;

2) условие прохождения сплайна через конечную точку (x_{i+1}, y_{i+1}) каждого участка;

3) условие гладкости функции в узлах интерполяции, т.е. непрерывности первой производной; математически это условие записывается как равенство первых производных в конце i-го участка и в начале $(i+1)$-го;

4) условие гладкости первой производной в узлах интерполяции, т.е. непрерывности второй производной, что означает равенство вторых производных в конце i-го участка и в начале $(i+1)$-го.

Учтем, что производные кубического сплайна записываются следующим образом:

$$\frac{dy}{dx} = 3a_i(x - x_i)^2 + 2b_i(x - x_i) + c_i \quad \text{и}$$

$$\frac{d^2y}{dx^2} = 6a_i(x - x_i) + 2b_i,$$

тогда можем записать перечисленные выше четыре условия сплайн-интерполяции для исходной функции с постоянным шагом h :

$$\begin{aligned}
&1)\ y_i = d_i\ , && i = 0,1,2,...,n-1\ , \\
&2)\ y_{i+1} = a_i h^3 + b_i h^2 + c_i h + d_i\ , && i = 0,1,2,...,n-1\ , \\
&3)\ 3a_i h^2 + 2b_i h + c_i = c_{i+1}\ , && i = 0,1,2,...,n-2\ , \\
&4)\ 6a_i h + 2b_i = 2b_{i+1}\ , && i = 0,1,2,...,n-2\ .
\end{aligned} \qquad (4.12)$$

Составим из этих четырех условий, записанных для каждого участка исходной функции, систему линейных уравнений, содержащую $(4n-2)$ уравнения с $4n$ неизвестными коэффициентами сплайнов a_i, b_i, c_i, d_i. Чтобы решить такую систему уравнений, необходимо добавить два дополнительных граничных условия. Граничные условия задают значения производных функции в конечных точках интервала и для сплайновой интерполяции могут иметь следующий вид:

$$y''(x_0) = y''(x_n) = 0 \ ,$$
$$y'(x_0) = y'(x_n) = 0 \ . \qquad (4.13)$$

Совместное решение $4n$ уравнений с $4n$ неизвестными позволяет найти коэффициенты кубических сплайнов на всех участках.

Вопросы для самоконтроля

1. Как можно повысить точность интерполяции?
2. Можно ли использовать метод Лагранжа для экстраполяции?
3. Как определить погрешность интерполяции в узле?
4. Можно ли в методе Ньютона выразить конечную разность только через исходные значения функции?
5. Сколько различных интерполяционных полиномов можно построить для n заданных точек?
6. Что называется кубическим сплайном?
7. Сколько коэффициентов, подлежащих определению, содержит кубический сплайн?
8. Какую функцию называют гладкой?
9. Могут ли узлы сплайнов располагаться неравномерно?
10. Как влияет количество узлов интерполяции на точность интерполяции?

ТЕМА 5. Аппроксимация

«Когда мы пытаемся из целого вытащить одно звено,
обнаруживается, что оно прочно связано со всем остальным»
Из законов Мерфи

5.1. Концепция аппроксимации

При наличии погрешности в исходных данных нецелесообразно применять для их анализа метод интерполяции и находить приближенную функцию, точно проходящую через все точки исходной таблично заданной функции. В таком случае прибегают к построению аппроксимирующей приближенной функции, которая проходит около заданных точек, причем наиболее близко к исходным точкам. Аппроксимация сглаживает обрабатываемые экспериментальные данные и позволяет, например, выделить полезный сигнал в потоке данных.

Методы аппроксимации применяют также к непрерывным функциям, когда есть необходимость получить упрощенное математическое описание сложной зависимости.

Пусть в результате измерений получена табличная зависимость величины y_i от величины x_i, которую можно графически представить набором точек на координатной плоскости (рис. 5.1). Если описать эту таблично заданную зависимость $y_i(x_i)$ специально подобранной аналитической функцией $\varphi(x)$, то можно вычислить приближенное значение в любой точке заданного интервала. При подборе аналитической функции следует учесть, что каждое измерение было выполнено с некоторой погрешностью. Поэтому правомерно разрешить отклонение измеренной величины от значения, вычисленного в этой же точке согласно аппроксимирующей функции: $\varepsilon_i = y_i(x_i) - \varphi(x_i)$.

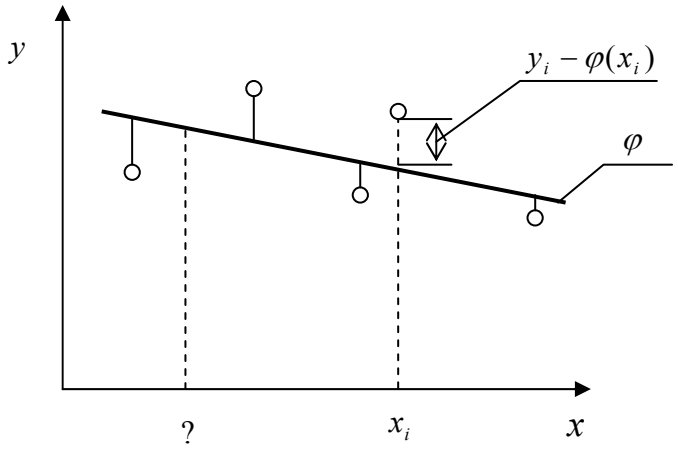


Рис.5.1. Исходные данные и аппроксимирующая функция

Близость исходной $y_i(x_i)$ и аппроксимирующей $\varphi(x)$ функций определяется некоторой числовой мерой, называемой критерием аппроксимации или критерием близости. Наибольшее распространение получил квадратичный критерий, равный сумме квадратов отклонений расчетных аппроксимирующих значений φ от исходных y_i для всех исходных точек x_i, в частности, экспериментальных значений, если проводится обработка результатов измерений (рис. 5.1). Итак, квадратичный критерий близости аппроксимирующей функции к исходной таблично заданной функции описывается выражением:

$$R = \sum_{i=1}^{n} \beta_i (y_i - \varphi(x_i))^2 , \qquad (5.1)$$

где y_i – заданные табличные значения функции, $\varphi(x_i)$ – расчетные значения аппроксимирующей функции в тех же точках, β_i – весовые коэффициенты, учитывающие относительную важность каждой исходной точки.

Чем больше значимость некоторой точки i в исходных данных для описания прогнозируемого явления, тем выше назначается ее весовой коэффициент.

Квадратичный критерий дифференцируем и обеспечивает единственное решение задачи аппроксимации для полиномиальных аппроксимирующих функций.

Различают два типа задач аппроксимации, когда при описании исходных данных необходимо:

1) получить аппроксимирующую функцию, описывающую имеющиеся данные с погрешностью не хуже заданной;

2) либо получить аппроксимирующую функцию заданной структуры с наилучшей возможной погрешностью.

Примером второго типа задач является аппроксимация результатов измерений кривой Гаусса (рис. 5.2). Рассеяние значений случайной величины, изменение которой зависит от большого числа факторов, когда ни один из факторов не имеет преобладающего влияния, подчиняется закону нормального распределения вероятностей (закону Гаусса):

$$y = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-a)^2}{2\sigma^2}}, \qquad (5.2)$$

где a – математическое ожидание случайной величины, σ – среднее квадратическое отклонение случайной величины.

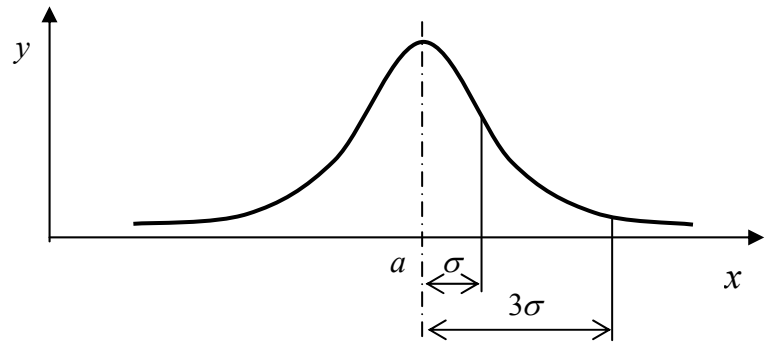


Рис. 5. 2. Кривая плотности вероятности нормального распределения

Закону Гаусса с некоторым приближением может подчиняться рассеяние погрешностей многократных измерений; рассеяние погрешностей изготовления; погрешности измерения линейных и угловых размеров, массы деталей, твердости и других механических и физических величин.

5.2. Метод наименьших квадратов

Метод наименьших квадратов базируется на применении в качестве критерия близости величины, равной сумме квадратов отклонений исходных и расчетных значений. Сумма квадратов отклонений выбирается в качестве критерия близости по аналогии с метрикой Евклидова пространства, где расстояние между точками определяется как длина вектора и выражается через сумму квадратов координат. Соответственно наименьший критерий близости при аппроксимации отражает минимальное пространственное отклонение аппроксимирующей функции от исходных данных. Если структура аппроксимирующей функции φ задана, то задача аппроксимации сводится к подбору таких параметров приближающей функции, которые обеспечивают наименьшее значение критерия близости, что собственно и соответствует наилучшей аппроксимации.

Рассмотрим в качестве примера полиномиальную аппроксимирующую функцию, которая в частном случае может вырождаться в линейную зависимость:

$$\varphi(x) = a_0 + a_1 x + ... + a_{k-1} x^{k-1} + a_k x^k = \sum_{j=0}^{k} a_j x^j \ . \qquad (5.3)$$

Пусть весовые коэффициенты исходных точек одинаковы и равны $\beta_i = 1$ при всех значениях индекса i, тогда критерий аппроксимации для аппроксимирующего полинома k-го порядка является функцией коэффициентов этого полинома и имеет вид

$$R = \sum_{i=1}^{n} (y_i - \sum_{j=0}^{k} a_j x_i^j)^2 = R(a_0, a_1, ..., a_{k-1}, a_{k-1}) \ . \qquad (5.4)$$

Необходимым условием решения поставленной задачи является сведение к минимуму критерия аппроксимации, следовательно, искомые параметры аппроксимирующей функции – коэффициенты полинома – можно определить, приравняв к нулю частные производные критерия по этим параметрам:

$$\frac{\partial R}{\partial a_m} = 2\sum_{i=1}^{n}(y_i - \sum_{j=0}^{k} a_j x_i^j)(-x_i^m) = 0 \ , \qquad (5.5)$$

где индекс $i = 1,2,...,n$ обозначает расчетные точки, индекс $j = 0,1,2,...,k$ обозначает коэффициенты полинома, индекс $m = 0,1,2,...,k$ обозначает текущий коэффициент полинома.

Полученные при дифференцировании равенства (5.5) можно преобразовать, а именно – разделить на два, раскрыть скобки, изменить порядок суммирования:

$$\sum_{i=1}^{n} y_i (-x_i^m) + \sum_{i=1}^{n}\sum_{j=0}^{k} a_j x_i^j x_i^m = \sum_{j=0}^{k} a_j \sum_{i=1}^{n} x_i^j x_i^m - \sum_{i=1}^{n} y_i x_i^m = 0 \ \text{ или}$$

$$\sum_{j=0}^{k} a_j \sum_{i=1}^{n} x_i^j x_i^m = \sum_{i=1}^{n} y_i x_i^m \ , \ j = 0,1,2,...,k \ , \ i = 1,2,...,n \ . \qquad (5.6)$$

Таким образом, для полинома k-го порядка мы получили систему из $k+1$ уравнений с таким же количеством неизвестных параметров a_j, причем линейную относительно этих параметров. Такая система называется системой нормальных уравнений. Из ее решения находятся коэффициенты a_j полинома – аппроксимирующей функции, обеспечивающие минимум критерия аппроксимации R, а значит, и наилучшее возможное квадратичное приближение.

Пример 5.1. К конструкции приложена переменная сила. С помощью датчика в контрольной точке фиксируется перемещение. Найти полиномиальную аппроксимирующую функцию для перемещения контрольной точки по имеющимся экспериментальным данным. Вычислить коэффициенты полиномов первого и второго порядка, построить графики.

Исходные данные приведены в таблице и представлены на рис. 5.3.

t, с	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
y, мм	10	15	18	14	7	3	6	9	10	8	5	2	1	2	4

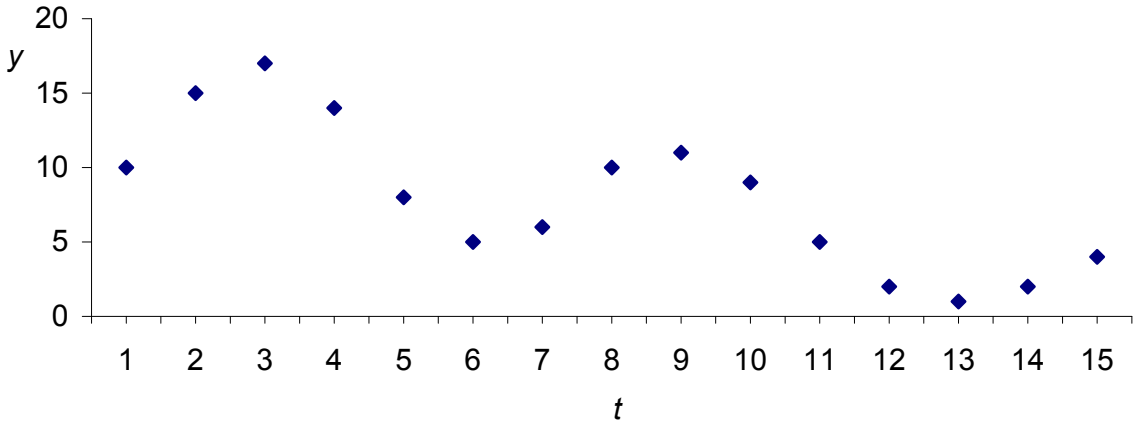


Рис. 5.3. Исходные данные

Решение. Число расчетных точек $n=15$.

Расчет 1. Аппроксимация линейным полиномом

$k=1$

Вычислим коэффициенты линейного полинома $\varphi = a_0 + a_1 t$. Система нормальных уравнений содержит два уравнения:

$$m=0\ ,\qquad t_i^{\,m}=1\ ,\qquad a_0\sum_{i=1}^{n}t_i^{\,0}+a_1\sum_{i=1}^{n}t_i=\sum_{i=1}^{n}y_i\ ,$$

$$m=1\ ,\qquad t_i^{\,m}=t_i\ ,\qquad a_0\sum_{i=1}^{n}t_i^{\,0}t_i+a_1\sum_{i=1}^{n}t_i^{\,2}=\sum_{i=1}^{n}y_i t_i\ ,$$

или
$$\begin{cases} a_0 n + a_1\sum\limits_{i=1}^{n}t_i=\sum\limits_{i=1}^{n}y_i \\ a_0\sum\limits_{i=1}^{n}t_i + a_1\sum\limits_{i=1}^{n}t_i^2=\sum\limits_{i=1}^{n}y_i t_i \end{cases} \qquad \begin{cases} a_0=\dfrac{\sum\limits_{i=1}^{n}t_i\sum\limits_{i=1}^{n}y_i t_i-\sum\limits_{i=1}^{n}t_i^2\sum\limits_{i=1}^{n}y_i}{\left(\sum\limits_{i=1}^{n}t_i\right)^2-n\sum\limits_{i=1}^{n}t_i^2} \\ a_1=\dfrac{\sum\limits_{i=1}^{n}y_i\sum\limits_{i=1}^{n}t_i-n\sum\limits_{i=1}^{n}y_i t_i}{\left(\sum\limits_{i=1}^{n}t_i\right)^2-n\sum\limits_{i=1}^{n}t_i^2} \end{cases}.$$

Вычислим суммы:

$$\sum_{i=1}^{15}t_i=120,\ \sum_{i=1}^{15}y_i=119,\ \sum_{i=1}^{15}t_i^{\,2}=1240,\ \sum_{i=1}^{15}y_i t_i=708\,.$$

Решив систему уравнений, получим $a_0=14.9$, $a_1=-0.87$. Соответствующий график линейной аппроксимирующей функции приведен на рис. 5.4.

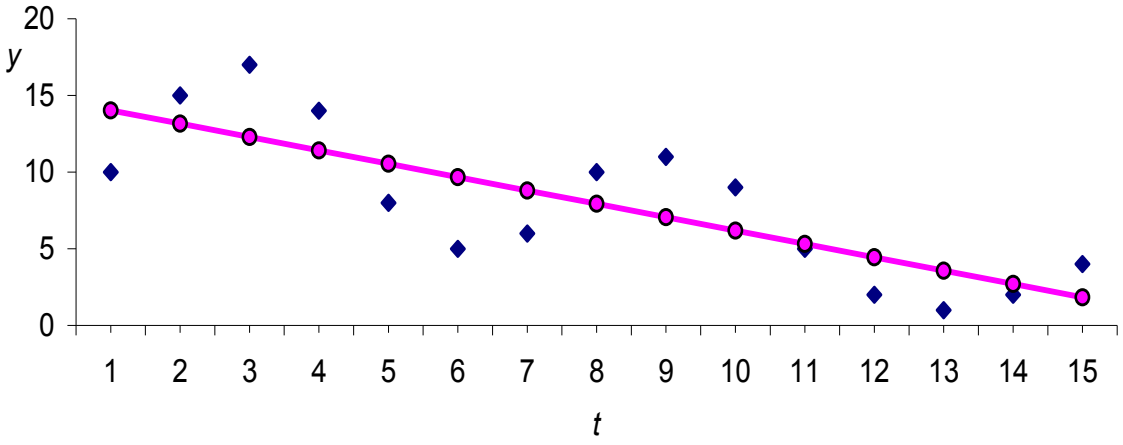


Рис. 5.4. Линейная аппроксимация исходной функции

Расчет 2. Аппроксимация квадратичным полиномом

$k = 2$

Вычислим коэффициенты полинома $\varphi = a_0 + a_1 t + a_2 t^2$. Система нормальных уравнений содержит три уравнения:

$$m = 0\ , \quad t_i^m = 1\ , \quad a_0 \sum_{i=1}^{n} t_i^0 + a_1 \sum_{i=1}^{n} t_i + a_2 \sum_{i=1}^{n} t_i^2 = \sum_{i=1}^{n} y_i\ ,$$

$$m = 1\ , \quad t_i^m = t_i\ , \quad a_0 \sum_{i=1}^{n} t_i^0 t_i + a_1 \sum_{i=1}^{n} t_i t_i + a_2 \sum_{i=1}^{n} t_i^2 t_i = \sum_{i=1}^{n} y_i t_i\ ,$$

$$m = 2\ , \quad t_i^m = t_i^2\ , \quad a_0 \sum_{i=1}^{n} t_i^0 t_i^2 + a_1 \sum_{i=1}^{n} t_i t_i^2 + a_2 \sum_{i=1}^{n} t_i^2 t_i^2 = \sum_{i=1}^{n} y_i t_i^2\ ,$$

или

$$\begin{cases} a_0 n + a_1 \sum_{i=1}^{n} t_i + a_2 \sum_{i=1}^{n} t_i^2 = \sum_{i=1}^{n} y_i \\ a_0 \sum_{i=1}^{n} t_i + a_1 \sum_{i=1}^{n} t_i^2 + a_2 \sum_{i=1}^{n} t_i^3 = \sum_{i=1}^{n} y_i t_i \\ a_0 \sum_{i=1}^{n} t_i^2 + a_1 \sum_{i=1}^{n} t_i^3 + a_2 \sum_{i=1}^{n} t_i^4 = \sum_{i=1}^{n} y_i t_i^2 \end{cases} .$$

Вычислим суммы $\sum_{i=1}^{15} t_i = 120$, $\sum_{i=1}^{15} y_i = 119$, $\sum_{i=1}^{15} t_i^2 = 1240$, $\sum_{i=1}^{15} y_i t_i = 708$,

$\sum_{i=1}^{15} t_i^3 = 14400$, $\sum_{i=1}^{15} t_i^4 = 178312$, $\sum_{i=1}^{15} y_i t_i^2 = 5906$.

Решив линейную систему уравнений, получим $a_0 = 14.95$, $a_1 = -0.88$, $a_2 = 0.0000004$. Соответствующий график аппроксимирующей функции приведен на рис. 5.5.

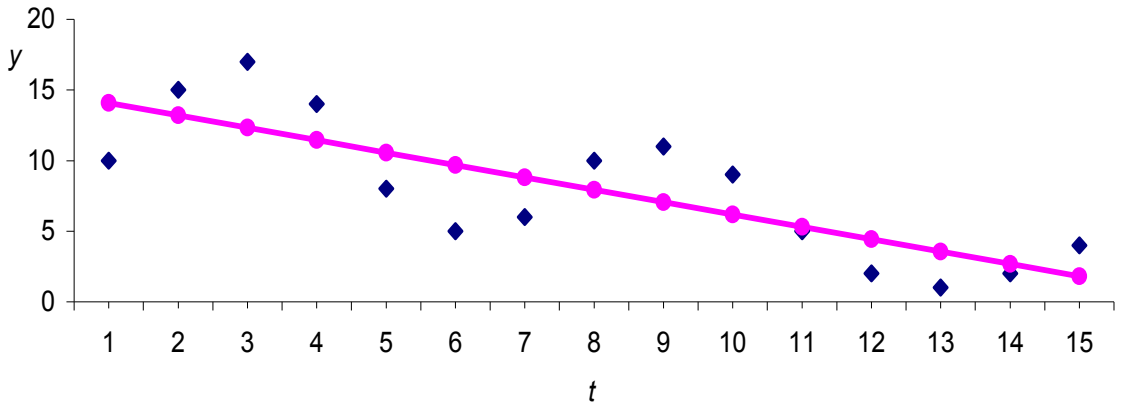


Рис. 5.5. Квадратичная аппроксимация исходной функции

□

Как видим, квадратичная аппроксимация мало отличается в приведенном примере от линейной, поскольку мал соответствующий коэффициент $a_2 = 0.0000004$ в полиноме. Исследуемая табличная функция требует аппроксимации полиномом более высокого порядка. Вычисление коэффициентов полинома более высокого порядка с применением описанного алгоритма представляет собой трудоемкую задачу. Для решения такой задачи можно использовать встроенные модули прикладных программ. Например, на рис. 5.6 представлена аппроксимация полиномом пятого порядка исходных данных, приведенных в примере 5.1.

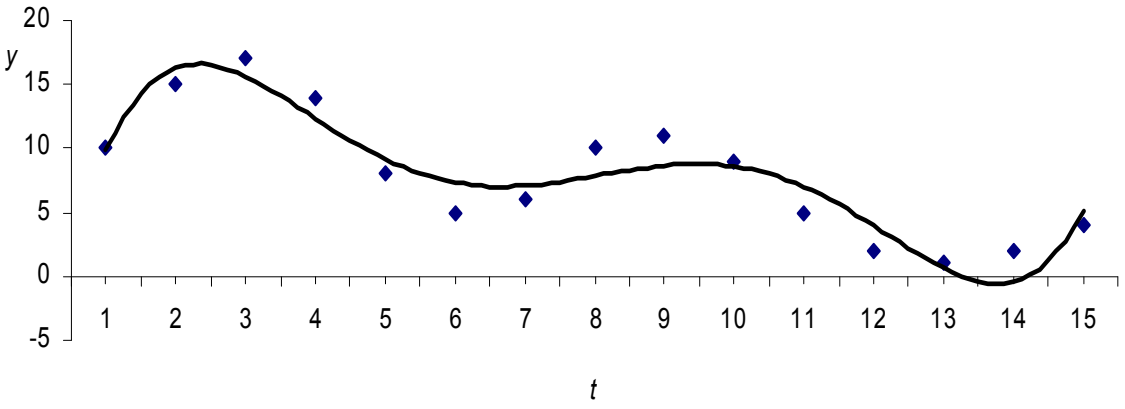


Рис. 5.6. Аппроксимация таблично заданной функции полиномом пятого порядка

5.3. Аппроксимация элементарными функциями

Аппроксимация дискретных данных, например последовательности показаний термопары с течением времени, не ограничивается линейной или полиномиальной зависимостью. В большинстве случаев для анализируемого процесса ожидается та функциональная зависимость, которая содержится в формулировке физического закона, описывающего этот процесс. Например, можно ожидать, что снижение температуры подчиняется экспоненциальному закону. Другой пример – при измерении предела выносливости получают последовательность значений предельного разрушающего напряжения в зависимости от количества циклов испытаний. Такая последовательность экспериментальных данных описывается логарифмической функцией. Соответственно чтобы найти предел выносливости как предельное значение последовательно представленных результатов измерения прочности образцов в зависимости от количества циклов нагружения, целесообразно аппроксимировать экспериментальные данные логарифмической функцией.

Для аппроксимации результатов измерений могут быть использованы любые элементарные функции: степенная, показательная, дробно-линейная, логарифмическая, гиперболическая, линейные комбинации перечисленных функций и функции произвольного вида. Причем аппроксимация табличных данных любой функцией, зависящей от двух коэффициентов, может быть сведена к нахождению линейной функции. Для всех нижеприведенных вариантов аппроксимации различными функциями продемонстрирован именно такой переход к нахождению коэффициентов линейной функции.

Степенная функция. Рассмотрим аппроксимирующую степенную функцию φ, которая в наиболее общем виде задается следующим выражением:

$$\varphi(x, a, m) = ax^m \ . \tag{5.7}$$

Предполагая, что все табличные значения аргумента и значения функции положительны, прологарифмируем обе части выражения при условии, что $a > 0$:

$$\ln\varphi = \ln a + m \ln x \ . \tag{5.8}$$

Так как функция φ является аппроксимирующей для таблично заданной функции y_i, то логарифмическая функция $\ln\varphi$ будет аппроксимирующей для функции $\ln y_i$. Введем новую переменную $u = \ln x$, тогда, как следует из предыдущего выражения, $\ln\varphi$ будет функцией от введенной переменной u: $\Phi(u) = \ln\varphi$.

Обозначим $m = A,\ \ln a = B$, тогда $\Phi(x, A, B) = Au + B$, следовательно, задача об аппроксимации исходной таблично заданной функции аналитической степенной функцией свелась к отысканию приближающей функции в виде линейной зависимости.

На практике для нахождения приближающей функции в виде степенной выполняют следующие действия. В исходной таблице данных логарифмируют все значения x_i и y_i; полученные величины помещают в новую таблицу. Затем вычисляют по новой таблице параметры A и B линейной приближающей функции. Вычисляют коэффициенты аппроксимирующей степенной функции a и m.

Показательная функция. Рассмотрим аппроксимирующую показательную функцию, которая в общем виде задается выражением:

$$\varphi(x, a, m) = ae^{mx},\ \ a > 0 \quad .$$

Прологарифмировав это равенство, получаем $\Phi(x, A, B) = Ax + B$.

Таким образом, аппроксимация показательной функцией сведена к поиску коэффициентов линейной функции.

Дробно-линейная функция. Рассмотрим аппроксимирующую дробно-линейную функцию, которая в общем виде задается следующим выражением:

$$\varphi(x, a, b) = \frac{1}{ax + b} \quad . \tag{5.9}$$

Обратим это выражение следующим образом:

$$\frac{1}{\varphi(x, a, b)} = ax + b \ . \tag{5.10}$$

Из последнего равенства следует, что необходимо задать новую табличную функцию, заменив значения исходной таблично заданной функции обратными числами. Далее, используя новую табличную функцию, следует

найти коэффициенты a и b, подставить их в начальное выражение дробно-линейной функции и найти значения аппроксимирующей функции.

Логарифмическая функция. Рассмотрим аппроксимирующую логарифмическую функцию, которая в общем виде задается выражением

$$\varphi(x,a,b) = a\ln x + b \quad . \tag{5.11}$$

Для перехода от логарифмической функции к линейной достаточно ввести новую переменную $u = \ln x$ и сделать подстановку. Следовательно, для нахождения коэффициентов a и b нужно прологарифмировать значения аргумента в исходной таблице данных. Далее, рассматривая полученные логарифмические значения в совокупности с исходными значениями функции, следует найти коэффициенты линейной функции a и b, подставить их в начальное выражение логарифмической функции и найти значения аппроксимирующей функции.

Гиперболическая функция. Рассмотрим аппроксимирующую гиперболическую функцию, которая в общем виде задается выражением

$$\varphi(x,a,b) = \frac{a}{x} + b \quad . \tag{5.12}$$

Для перехода к линейной функции достаточно ввести новую переменную $u = \dfrac{1}{x}$ и сделать подстановку, тогда $\varphi(x,a,b) = \varphi(u,a,b) = au + b$.

Последовательность вычисления коэффициентов включает замену значений аргумента обратными числами и поиск коэффициентов линейной функции.

Дробно-рациональная функция. Рассмотрим аппроксимирующую дробно-рациональную функцию, которая задается выражением:

$$\varphi(x,a,b) = \frac{x}{ax + b} \quad . \tag{5.13}$$

Произведем замену дроби: $\dfrac{1}{\varphi(x,a,b)} = a + \dfrac{b}{x}$. Далее решение задачи аналогично аппроксимации гиперболической функцией. Введем замену: $u = \dfrac{1}{x}$, $\Phi = \dfrac{1}{y}$. В таблице исходных данных заменяем значения функции и

аргумента на обратные числа и находим коэффициенты линейной зависимости $\Phi(u,a,b) = bu + a$.

Аппроксимация линейной комбинацией функций. Будем искать аппроксимирующую функцию $\varphi(x)$ в виде линейной комбинации элементарных функций $f_1(x),\ f_2(x),\ f_3(x)$:

$$\varphi(x) = af_1(x) + bf_2(x) + cf_3(x). \tag{5.14}$$

Применив метод наименьших квадратов, получим систему линейных уравнений относительно неизвестных коэффициентов a,b,c. Такая система уравнений в матричной записи имеет вид:

$$\begin{pmatrix} \sum f_1(x_i)^2 & \sum f_1(x_i)f_2(x_i) & \sum f_1(x_i)f_3(x_i) \\ \sum f_1(x_i)f_2(x_i) & \sum f_2(x_i)^2 & \sum f_2(x_i)f_3(x_i) \\ \sum f_1(x_i)f_3(x_i) & \sum f_2(x_i)f_3(x_i) & \sum f_3(x_i)^2 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} \sum y_i f_1(x_i) \\ \sum y_i f_2(x_i) \\ \sum y_i f_3(x_i) \end{pmatrix} \tag{5.15}$$

Решив полученную систему линейных уравнений любым известным способом, можно найти коэффициенты a,b,c и тем самым однозначно определить аналитическое выражение для аппроксимирующей функции.

Вопросы для самоконтроля

1. Можно ли при аппроксимации полиномом таблично заданной функции обеспечить прохождении аппроксимирующей функции точно через все точки?
2. Каково назначение весовых коэффициентов в критерии близости исходной и аппроксимирующей функций?
3. Можно ли повысить точность, одновременно увеличив в несколько раз все весовые коэффициенты?
4. Можно ли с помощью метода наименьших квадратов найти параметры неполиномиальной аппроксимирующей функции?
5. Может ли степень аппроксимирующего полинома быть выше числа узлов аппроксимации?

ТЕМА 6. Нелинейные уравнения

«Есть правила для выбора решения,
но нет правил для выбора этих правил»
Из законов Мерфи

6.1. Концепция методов решения нелинейных уравнений

Если законы функционирования некоторого инженерного объекта являются нелинейными, а моделируемые процессы или системы обладают одной степенью свободы (т.е. имеют одну независимую переменную), то модель объекта, как правило, описывается одним нелинейным уравнением. В общем случае нелинейное уравнение с одним неизвестным можно записать в виде $f(x)=0$, где $f(x)$ – некоторая непрерывная функция аргумента x.

Необходимость отыскания корней нелинейных уравнений встречается в расчетах систем автоматического управления и регулирования, собственных колебаний машин и конструкций, в задачах кинематического анализа и синтеза, плоских и пространственных механизмов и других задачах.

Как правило, численное решение нелинейного уравнения общего вида $f(x)=0$ осуществляется в два этапа. На первом этапе отделяют корни, т.е. находят такие отрезки на числовой прямой, внутри каждого из которых находится один корень. На втором этапе для каждого выделенного интервала уточняют корень, т.е. находят его приближенное значение x_r с предварительно заданной точностью ε.

Различают алгебраические и трансцендентные уравнения. Алгебраические уравнения путем алгебраических преобразований можно привести к каноническому виду

$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0 = 0 \ , \qquad (6.1)$$

где $a_0, a_1, \ldots, a_n$ – коэффициенты уравнения. Показатель n называют степенью алгебраического уравнения.

Если функция $f(x)$ не является алгебраической, уравнение $f(x)=0$ называется трансцендентным. Пример трансцендентного уравнения: $2^x + \cos(x) = 0$. Трансцендентные уравнения могут иметь бесконечное

множество решений. В некоторых случаях решение трансцендентных уравнений можно свести к решению алгебраических уравнений.

Методы решения нелинейных уравнений делятся на *прямые* и *итерационные*. Прямые методы позволяют записать решение в виде некоторого соотношения, формулы. При этом значения корней могут быть вычислены по этой формуле за конечное число арифметических операций. Подобные методы развиты для решения тригонометрических, логарифмических, показательных, а также простейших алгебраических уравнений.

Однако подавляющее большинство нелинейных уравнений, встречающихся на практике, не удается решить прямыми методами. Во всех таких случаях приходится обращаться к численным методам, позволяющим получить приближенные значения корней с заранее заданной точностью.

Идея нахождения приближенных значений корней нелинейных уравнений состоит в следующем. Прежде всего, некоторым образом выбирают *начальное приближение* к корню $x_0 \in [a,b]$. На основе этого значения по некоторой формуле вычисляют следующее приближение x_1, затем x_2 и т.д. Каждый такой шаг называется *итерацией* (от латинского *iteratio* – повторение), а сами методы уточнения – *итерационными методами*. В результате итераций получается последовательность приближенных значений корня $x_0, x_1, ..., x_k, ...$, которая называется *итерационной последовательностью*.

Первый этап: отделение корней. Все методы отделения корней (методы локализации) базируются на очевидном свойстве непрерывных функций: корни уравнения $f(x) = 0$ задаются точками пересечения графика функции $f(x)$ с осью абсцисс $y = 0$ и находятся между соседними экстремумами функции.

Отделение корней можно провести графически путем построения графика функции $f(x)$ и выявления точек ее пересечения с осью абсцисс. В случае аналитического отделения корней находят все критические точки функции $f(x)$, т.е. точки, в которых производные равны нулю или не существуют (например, знаменатель обращается в нуль). Для этого необходимо дифференцировать функцию $f(x)$.

В найденных критических точках или в непосредственной близости от них определяют знак функции $sign\ f(x_i)$, включая знак функции на бесконечном удалении числовой оси $-\infty$ и $+\infty$. Полученные данные ана-

лизируют: число смен знаков $sign\ f(x_i)$ равно количеству корней. Причем каждый корень локализован в таком интервале, где на левой границе интервала и на правой его границе функция $f(x)$ имеет разные знаки. Выявленные интервалы локализации корней можно сузить, используя дополнительные точки, которые заменяют границы в бесконечности.

Пример 6.1. Дано уравнение $\frac{2}{3}x^3 - \frac{1}{2}x^2 - 6x + \frac{1}{3} = 0$; провести аналитическое отделение корней уравнения.

Решение. Анализируемая функция является полиномом третьей степени:

$$f(x) = \frac{2}{3}x^3 - \frac{1}{2}x^2 - 6x + \frac{1}{3},$$

ее производная также является полиномом: $\frac{df}{dx} = 2x^2 - x - 6$.

Найдем корни уравнения $\frac{df}{dx} = 0$; $2x^2 - x - 6 = 0$; преобразуем уравнение: $(x-2)(2x+3) = 0$; корни уравнения: $x_1 = 2$; $x_2 = -\frac{3}{2}$.

Представим информацию об изменении знака функции $f(x)$ на всей числовой оси в виде таблицы:

x	$-\infty$	-3/2	2	$+\infty$
$sign\ f(x)$	-	+	-	+

Из таблицы видно, что уравнение $f(x) = 0$ имеет три действительных корня: $x_1 \in \left]-\infty, -\frac{3}{2}\right[$; $x_2 \in \left[-\frac{3}{2}, 2\right]$; $x_3 \in]2, +\infty[$. Уменьшим промежутки, в которых находятся корни:

x	$-\infty$	-3	-3/2	-1	1	2	4	$+\infty$
$sign\ f(x)$	-	-	+	+	-	-	+	+

Окончательно получаем, что корни уравнения находятся внутри интервалов $x_1 \in \left[-3,-\frac{3}{2}\right]$, $x_2 \in [-1,1]$, $x_3 \in [2,4]$.

□

Второй этап: уточнение корней. Уточнение корней можно проводить одним из трех следующих способов, различающихся концептуально.

Первый способ: поиск корня с заданной погрешностью сводится к перебору всех возможных значений аргумента. В каждой текущей точке проверяют наличие решения; такой поиск называют сканированием.

Второй способ: поиск корня нелинейного уравнения заменяется поиском корня более простого уравнения. Преимущественно выбирается линейная или параболическая функция. Поиск решения осуществляется посредством выполнения однотипных повторяющихся вычислений.

Третий способ: нелинейное уравнение $f(x)=0$ сводят к виду $g(x)=\varphi(x)$ и стремятся обеспечить равенство левой и правой частей с помощью итерационных процедур.

Точное решение уравнения не всегда является необходимым. Задачу отыскания корней уравнения можно считать практически решенной, если мы сумеем найти корни уравнения с заданной степенью точности. Считается, что уравнение решено и его корень x_r с заданной погрешностью ε найден, если выполняется условие $|f(x_r)| \le \delta$ или $|x_r - x_k| \le \varepsilon$, где δ, ε – предварительно заданные малые положительные величины, k – номер итерации. Согласно сформулированным условиям поиск корня методом итераций или методом сканирования прерывается в тот момент, когда близка к нулю левая часть уравнения или близки друг к другу два текущих значения x, между которыми находится решение.

При прочих равных условиях более эффективным считается такой метод уточнения корней, который позволяет найти решение с той же погрешностью за меньшее количество вычислений.

6.2. Метод сканирования

Метод сканирования предусматривает разделение всего интервала $[a,b]$, где отделен корень, на маленькие отрезки, равные заданной погрешности ε, с последующим вычислением значений функции $f(x_i)$ на концах этих отрезков, т.е. в точках, расстояние между которыми не превышает ве-

личины погрешности (рис. 6.1). По значениям функции, вычисленным с шагом ε, выбирают тот отрезок $[x_i, x_{i+1}]$, где функция меняет знак.

За корень уравнения принимают середину выбранного отрезка:

$$x_r = \frac{x_i + x_{i+1}}{2} \ . \qquad (6.2)$$

Исходя из описанного алгоритма поиска корней уравнения, погрешность решения не будет превышать заданную погрешность ε.

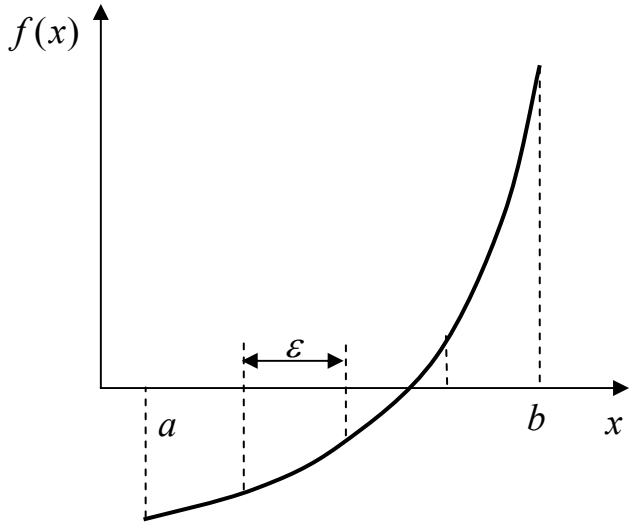


Рис. 6.1. Поиск корня методом сканирования

Объем вычислений при использовании метода сканирования можно уменьшить за счет разбиения интервала на отрезки с удвоенной длиной 2ε. Другой способ повысить эффективность метода – проводить уточнение поэтапно с переменным шагом разбиения. На первом этапе задается большое значение шага и находится тот отрезок, где функция меняет знак. Затем найденный отрезок еще раз делится с меньшим шагом, корень находится более точно; и так далее.

6.3. Метод деления отрезка пополам

Метод деления отрезка пополам предполагает деление интервала на две равные части, затем сравнивают знаки функции на концах каждой из двух половинок и выделяют ту половину, на концах которой знаки функции разные (рис. 6.2). Выделенную половину снова делят на две равные части, снова выбирают одну из половинок, содержащую корень, и т.д.

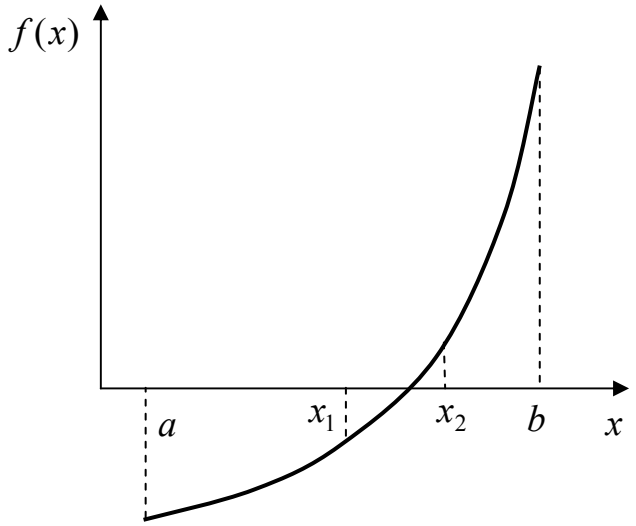


Рис. 6.2. Поиск корня методом половинного деления

Условием окончания итерационного процесса служит деление текущего отрезка на такие половинки, длины которых не превышают заданной погрешности ε.

6.4. Метод хорд

Метод хорд предполагает, что на отделенном интервале $[a,b]$ нелинейная функция $f(x)$ заменяется линейной функцией. Заменяющая линейная функция отражается на графике хордой – отрезком, соединяющим граничные точки $(a,f(a))$ и $(b,f(b))$ нелинейной функции.

По линейному уравнению хорды можно найти точку ее пересечения с осью абсцисс и, соответственно, приближенное решение x_1 на первом шаге вычислений. Найденная точка принимается за новую границу отрезка, в котором содержится корень. Через эту точку с координатами $(x_1, f(x_1))$ и предварительно зафиксированную границу начального интервала снова проводят хорду, находят следующее приближение x_2, и т.д.

В результате получается последовательность значений $x_1, x_2, x_3, ..., x_i$, сходящаяся к корню (рис. 6.3). Условием прекращения вычислений может быть неравенство $|x_i - x_{i-1}| \le \varepsilon$, а в качестве корня уравнения с точностью ε может быть принято значение x_i.

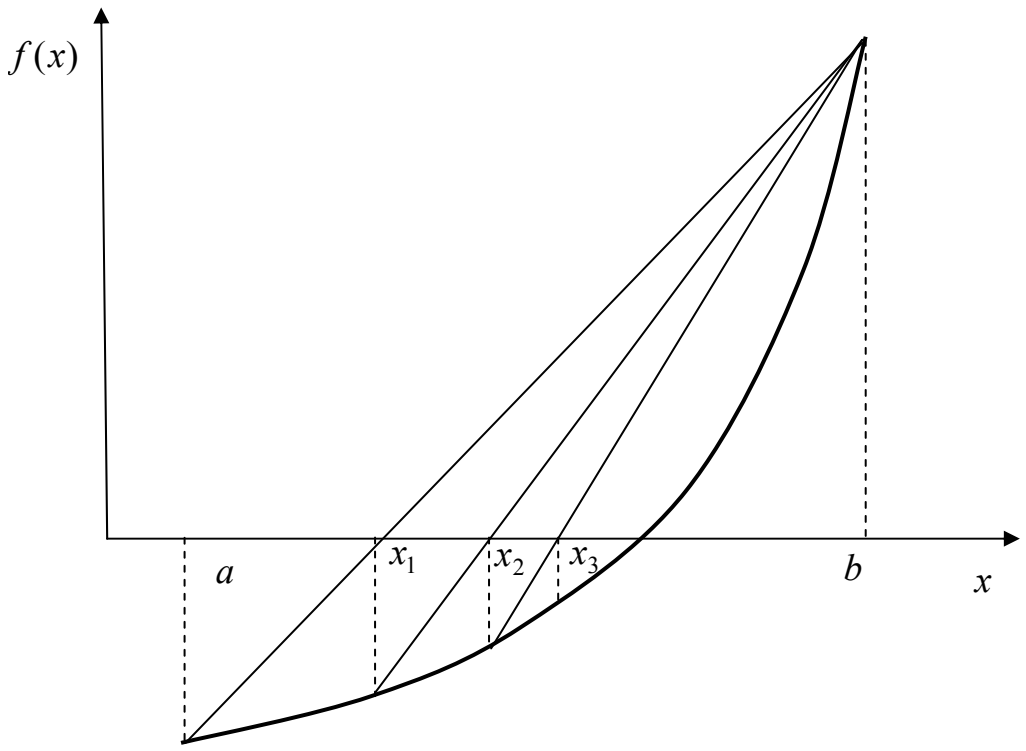


Рис. 6.3. Поиск корня методом хорд

Метод применим к монотонной на заданном участке функции. Фиксируется при этом правая или левая граница интервала в зависимости от вида функции.

Если $f(b)\cdot f''(b)>0$, то фиксируется правая граница интервала (рис. 6.3) и вычисление производится по формуле

$$x_{i+1}=x_i-\frac{f(x_i)}{f(b)-f(x_i)}(b-x_i)\ ; \qquad (6.3)$$

при этом последовательность $x_1,x_2,x_3,...,x_n$ приближается к корню слева.

Если $f(a)\cdot f''(a)>0$, то фиксируется левая граница интервала и вычисление производится по формуле

$$x_{i+1}=a+\frac{f(a)}{f(a)-f(x_i)}(x_i-a)\ ; \qquad (6.4)$$

при этом последовательность найденных решений $x_1,x_2,x_3,...,x_n$ приближается к корню справа.

6.5. Метод касательных

Метод касательных (Ньютона) использует в качестве отсекающей прямой касательную к графику функции в текущей точке итерационной последовательности (рис. 6.4).

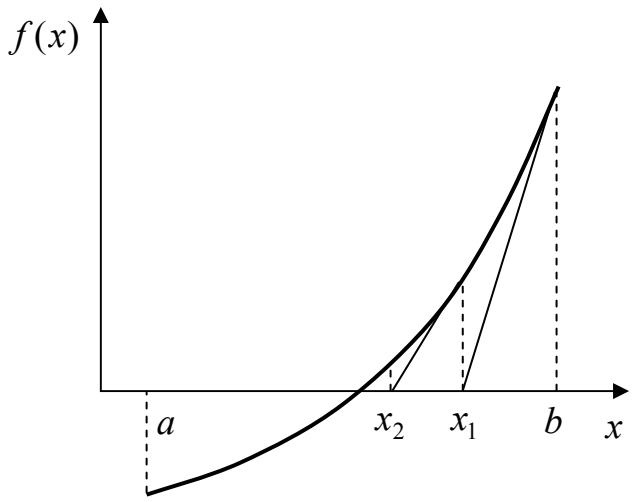


Рис. 6.4. Поиск корня методом касательных

Уравнение касательной, определяющее первую производную, задается по координате текущей точки и углу наклона касательной в этой точке. В качестве начальной точки можно выбрать левую ($x_0 = a$) или правую ($x_0 = b$) границу интервала. Левая точка интервала выбирается в том случае, если выполняется неравенство $f(a) \cdot f''(a) > 0$; соответственно правая точка выбирается в том случае, если $f(b) \cdot f''(b) > 0$. Алгоритм итерационных вычислений записывается следующим образом:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \ . \qquad (6.5)$$

Алгоритм надежно работает, когда на выделенном интервале анализируемая функция $f(x)$ является монотонной. Главным вычислительным достоинством метода является квадратичная скорость сходимости, что во многих случаях может привести к сокращению объема вычислений. Условие окончания поиска аналогично методу хорд.

6.6. Метод параболической аппроксимации

Метод параболической аппроксимации заменяет исходную функцию параболической функцией (рис. 6.5). Такая аппроксимация обеспечивает более быструю сходимость к решению.

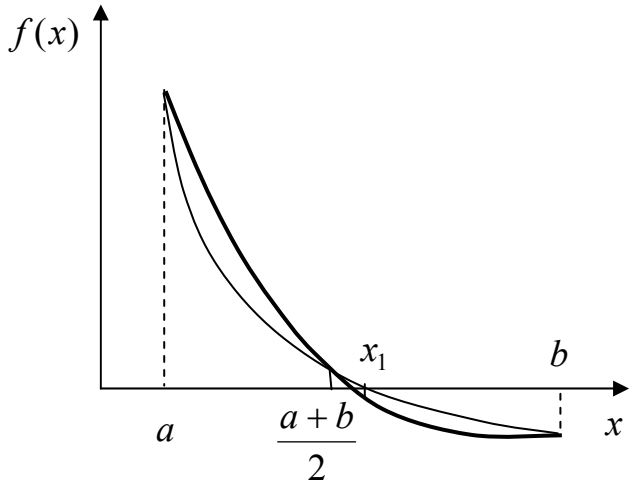


Рис. 6.5. Иллюстрация первого этапа
при поиске корня методом параболической аппроксимации

На первом этапе для построения параболы выбирают три точки: две граничные точки $(a, f(a))$, $(b, f(b))$ и срединную точку ограниченного интервала с координатами $((a+b)/2, f((a+b)/2))$. По уравнению параболы $y = c_2 x^2 + c_1 x + c_0$ находят приближенный корень, решая уравнение $c_2 x^2 + c_1 x + c_0 = 0$.

На втором этапе для построения параболы используют три точки: $((a+b)/2, f((a+b)/2))$, $(x_1, f(x_1))$, $(b, f(b))$. На третьем этапе для построения параболы будут использованы следующие три точки: $((a+b)/2, f((a+b)/2))$, $(x_2, f(x_2))$, $(x_1, f(x_1))$. Далее для построения парабол будет использована тройка предыдущих расчетных значений: $(x_2, f(x_2))$, $(x_3, f(x_3))$, $(x_1, f(x_1))$.

Такая процедура повторяется многократно до тех пор, пока величина отрезка, внутри которого находится корень, не станет меньше предварительно заданной погрешности.

6.7. Метод простой итерации

Метод простой итерации требует преобразования исходного уравнения $f(x)=0$ к виду

$$\varphi(x)=x \tag{6.6}$$

или в общем случае – к виду $f(x)=g(x)$. Существуют различные способы преобразования исходного уравнения к виду (6.6). Самый простой способ заключается в следующей замене:

$$f(x)=0 \qquad \rightarrow \begin{cases} f(x)+x=0+x \\ f(x)+x=\varphi(x) \end{cases} \qquad \rightarrow \varphi(x)=x \ . \tag{6.7}$$

На начальном этапе переменной присваивают некоторое произвольное значение x_0, которое не является корнем уравнения, и соответственно $\varphi(x_0)\neq x_0$. На первом этапе вычисляют значение $x_1=\varphi(x_0)$, которое является первым приближением к корню. Далее вычисляют следующее значение $x_2=\varphi(x_1)$ и т.д. В общем случае на некотором итерационном шаге приближенное решение исходного уравнения вычисляется как $x_{i+1}=\varphi(x_i)$. Условием окончания поиска может быть неравенство: $|x_i - x_{i+1}|\le\varepsilon$, где ε – заданная погрешность решения.

Полученная последовательность значений $x_0, x_1, x_2, ..., x_{i+1}, ...$ сходится к корню (рис. 6.6) при условии, что модуль производной заменяющей функции не превышает единицу: $|\varphi'(x)|\le 1$ на интервале $[a,b]$. Причем чем ближе модуль к нулю, тем выше скорость сходимости к решению.

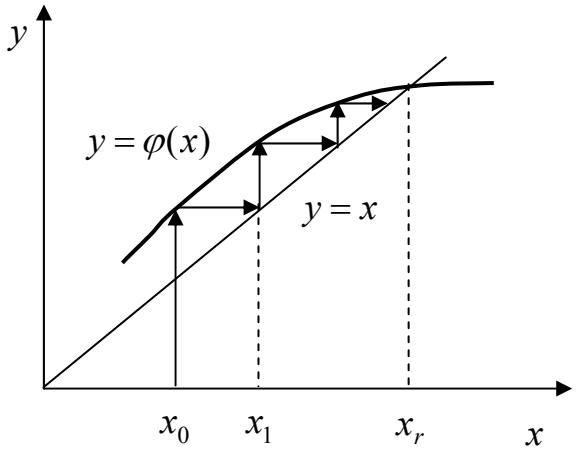


Рис. 6.6. Поиск корня методом простой итерации

Если условие $|\varphi'(x)| \le 1$ не выполняется, итерационная последовательность не сходится к искомому решению (рис. 6.7).

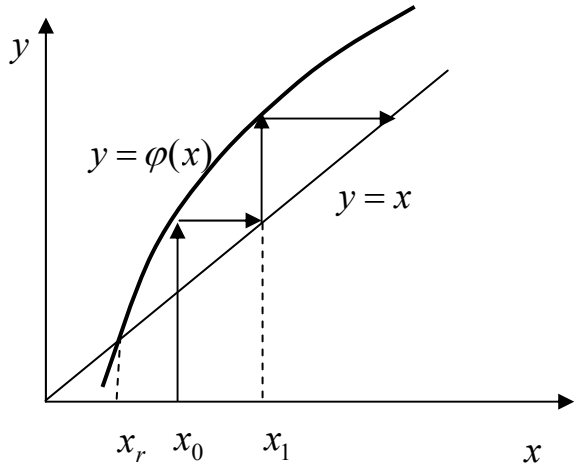


Рис. 6.7. Несходящийся итерационный процесс

Решение нелинейных уравнений с одним неизвестным является важной задачей компьютерного инжиниринга. Как правило, при решении научных и инженерных задач функция $f(x)$ содержит ряд параметров, исследователя и инженера интересует поведение решений уравнения $f(x, p_1, p_2, ...p_k) = 0$ в зависимости от параметров p_k. Не нарушая общности задачи, можно поменять местами неизвестное x и любой из параметров p_k, т.е. решить уравнение относительно другой неизвестной величины.

Вопросы для самоконтроля

1. Можно ли найти корень методом деления отрезка пополам, если он находится на границе интервала?
2. Всегда ли метод хорд позволяет вычислить отделенный корень с заданной погрешностью?
3. В чем заключается геометрическая интерпретация метода Ньютона?
4. В каких случаях метод параболической аппроксимации не найдет корень уравнения?
5. Что называется сходимостью метода итераций?
6. Если на заданном отрезке имеется два корня, то что можно сказать о сходимости метода итераций на этом отрезке?

ТЕМА 7. Системы линейных уравнений

«Никогда не ставьте задачу,
решение которой вам неизвестно»
Из законов Мерфи

7.1. Концепция методов решения систем линейных уравнений

Математические модели многих инженерных задач приводят к системам линейных алгебраических уравнений с большими и разреженными матрицами коэффициентов. В компьютерном инжиниринге нас прежде всего интересует решение уравнений с частными производными методами конечных разностей или конечных элементов.

Решение систем линейных уравнений в компьютерном инжиниринге проводится численными методами линейной алгебры. В линейной алгебре рассматриваются четыре класса задач: решение систем линейных алгебраических уравнений (СЛАУ), вычисление определителей, нахождение обратных матриц, определение собственных векторов и собственных значений матриц. Все эти задачи имеют важное прикладное значение при решении различных научно-технических задач в компьютерном инжиниринге. Кроме того, задачи линейной алгебры являются вспомогательными при реализации многих алгоритмов вычислительной математики, математической физики, обработки результатов экспериментальных исследований.

Методы решения СЛАУ можно разделить на две большие группы – прямые и итерационные. Прямые методы позволяют найти решение системы в результате однократного применения вычислительной процедуры. В итерационных методах вычислительная процедура применяется многократно. Применение итерационных методов целесообразно в тех случаях, когда система содержит большое количество линейных уравнений.

В общем виде система линейных алгебраических уравнений записывается следующим образом:

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_1 \,, \\ &a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2n}x_n = b_2 \,, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots + a_{nn}x_n = b_n \,. \end{aligned} \qquad (7.1)$$

Системы уравнений для численного решения записывают в матричном виде

$$\begin{bmatrix} a_{11} & a_{12} & .. & a_{1n} \\ a_{21} & a_{22} & .. & a_{2n} \\ .. & .. & .. & .. \\ a_{n1} & a_{n2} & .. & a_{nn} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ .. \\ x_n \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ .. \\ b_n \end{Bmatrix}$$

или компактно: $[A]\{x\} = \{B\}$, где $[A]$ – квадратная матрица размером $n \times n$, составленная из коэффициентов при неизвестных; n – размерность системы уравнений; $\{x\}$ и $\{B\}$ – матрицы размером $n \times 1$, содержащие неизвестные и свободные члены соответственно. Система уравнений имеет решение, если число неизвестных равно количеству уравнений, поэтому матрица коэффициентов должна быть квадратной: количество столбцов матрицы коэффициентов равно числу неизвестных, а количество строк равно количеству уравнений.

Матрицы – полезный аппарат для исследования многих задач математики, физики и техники. Одной из важнейших задач является поиск решения систем линейных алгебраических уравнений.

7.2. Матрицы и матричные вычисления

Многие технические задачи в приближенных расчетах сводятся к решению систем линейных уравнений в матричном виде, кроме того, при решении инженерных задач исходные и обрабатываемые табличные данные могут быть представлены для компьютерных вычислений в виде матриц A размера $m \times n$:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}, \qquad (7.2)$$

где числа a_{ij}, составляющие данную матрицу, называются элементами матрицы; первый индекс i в обозначении элемента матрицы указывает на номер строки, второй индекс j – на номер столбца.

Матрица, у которой число строк равно числу столбцов, называется квадратной. Матрица, содержащая один столбец или одну строку, называ-

ется *вектором* (вектор-столбец или вектор-строка соответственно). Вектор-столбец и вектор-строка представляются следующим образом:

$$A = \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_m \end{pmatrix}, \quad B = \begin{pmatrix} b_1 & b_2 & \dots & b_n \end{pmatrix}. \tag{7.3}$$

Элементы a_{11}, a_{22}, …, a_{ii} образуют главную диагональ матрицы. Квадратная матрица, у которой все элементы, кроме элементов главной диагонали, равны нулю, называется *диагональной*. Диагональная матрица, у которой каждый элемент главной диагонали равен единице, называется *единичной*:

$$E = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{pmatrix}. \tag{7.4}$$

Матрица, полученная из исходной путем замены каждой ее строки на столбец с тем же номером, называется матрицей, *транспонированной* к данной и обозначается A^T. Например, если $A = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, то $A^T = \begin{pmatrix} 1 & 0 \end{pmatrix}$.

При сложении матриц A и B получают матрицу $C = A + B$, при этом складываются их элементы с одинаковыми индексами: $c_{ij} = a_{ij} + b_{ij}$. Сложение проводится только для матриц одинаковых размеров.

При умножении матрицы A на число k получают матрицу $C = kA$, при этом каждый элемент исходной матрицы умножается на данное число: $c_{ij} = ka_{ij}$.

Произведением матриц A и B является матрица C, каждый элемент которой вычисляется как сумма произведений элементов i-й строки матрицы A на соответствующие элементы k-го столбца матрицы B: $c_{ik} = a_{i1}b_{1k} + a_{i2}b_{2k} + \dots + a_{in}b_{nk}$. Операция умножения проводится для двух матриц, когда число столбцов первой матрицы равно числу строк второй матрицы. Последовательность вычислений при умножении матриц можно представить схемой (рис. 7.1).

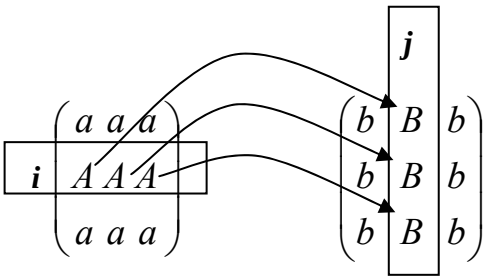


Рис. 7.1. Схема умножения матриц

Произведение двух квадратных матриц одинакового порядка не обладает в общем случае перестановочным свойством: $AB \neq BA$. Простой пример: если $A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ и $B = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$, то $AB = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$ и $BA = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$.

Матрицы, для произведения которых справедливо перестановочное свойство, называют *коммутирующими* матрицами. Следует отметить особую роль единичной матрицы *E*, аналогичную той роли, которую играет число 1 при перемножении вещественных чисел: при умножении произвольной квадратной матрицы на единичную получают исходную матрицу: $AE = EA = A$.

Эквивалентные матрицы – матрицы, которые могут быть получены одна из другой с помощью элементарных преобразований, а именно:

1) перестановкой местами двух строк матрицы;

2) умножением всех элементов строки на число, отличное от нуля;

3) сложением двух строк.

Определитель матрицы или ее детерминант – вещественное число, вычисляемое по элементам матрицы согласно следующим правилам.

Для квадратной матрицы второго порядка

$\det A = a_{11}a_{22} - a_{12}a_{21}$.

Для квадратной матрицы третьего порядка

$\det A = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{21}a_{32}a_{13} - a_{31}a_{22}a_{13} - a_{21}a_{12}a_{33} - a_{32}a_{23}a_{11}$.

Определитель матрицы более высокого порядка вычисляется через определители этой матрицы низших порядков.

Минор m_{ij} элемента a_{ij} определителя матрицы – определитель, полученный из исходного после вычеркивания строки и столбца, на пересечении которых находится данный элемент.

Алгебраическое дополнение A_{ij} элемента a_{ij} определителя матрицы удовлетворяет условию: $A_{ij} = (-1)^{i+j} m_{ij}$.

Присоединенная матрица A^* имеет элементы, равные алгебраическим дополнениям A_{ij} элементов a_{ij} исходной матрицы:

$$A^* = \begin{pmatrix} A_{11} & A_{21} & \dots & A_{n1} \\ A_{12} & A_{22} & \dots & A_{n2} \\ \dots & \dots & \dots & \dots \\ A_{1n} & A_{2n} & \dots & A_{nn} \end{pmatrix}.$$

Обратная матрица – квадратная матрица того же порядка, произведение которой на исходную матрицу равно единичной матрице:

$$AA^{-1} = A^{-1}A = E \ . \tag{7.5}$$

Обратная матрица находится делением присоединенной матрицы на определитель исходной матрицы:

$$A^{-1} = \frac{A^*}{\det A} \ . \tag{7.6}$$

Процедура нахождения обратной матрицы используется при решении системы линейных алгебраических уравнений. Систему уравнений можно представить в матричном виде: $[A]\{x\} = \{B\}$, где $\{x\}$ – искомое решение системы, записанное в виде вектора-столбца. Решение этого матричного уравнения можно свести к нахождению обратной матрицы $[A]^{-1}$, поскольку, умножив слева обе части уравнения на обратную матрицу $[A]^{-1}[A]\{x\} = [A]^{-1}\{B\}$, получаем непосредственно вектор-столбец неизвестных значений: $\{x\} = [A]^{-1}\{B\}$.

Вырожденная матрица – матрица, определитель которой равен нулю. Всякая невырожденная матрица имеет обратную матрицу.

7.3. Матрицы, обрабатываемые CAE-программами

Программы инженерного анализа CAE формируют и обрабатывают матрицы большой размерности, особым образом структурированные.

Разреженная матрица – матрица, большинство элементов которой равно нулю. Вводится понятие плотности матрицы – отношение числа не-

нулевых элементов к общему числу элементов матрицы. Матрица считается разреженной, если ее плотность меньше $\frac{1}{\sqrt{N}}$, где N – размерность матрицы. Например, если размерность матрицы $N = 10^3$, общее число элементов матрицы равно 10^6, тогда число ненулевых элементов разреженной матрицы не превышает 31622.

Разреженные матрицы формируются при переходе к компьютерному решению различных задач с помощью программ компьютерного инженерного анализа. В процессе построения дискретных аналогов таких задач возникают большие системы линейных алгебраических уравнений, представляемые в компактной записи матричными уравнениями:

$$[A]\{x\} = \{f\}\ , \tag{7.7}$$

где $\{x\}$ – вектор искомых значений некоторой физической величины (температура, смещение и т.п.) в расчетных точках дискретной модели;
$\{f\}$ – вектор нагрузок в расчетных точках дискретной модели.

Матрицы $[A]$ в таких уравнениях, как правило, симметричны относительно главной диагонали и имеют разреженную структуру. При решении задач на прочность и жесткость конструкций матрица $[A]$ называется глобальной матрицей жесткости системы, в тепловых задачах матрица $[A]$ имеет смысл теплового сопротивления.

Ленточная матрица – разреженная матрица с ленточной структурой ненулевых элементов. Ленточные матрицы формируются при составлении глобальной матрицы жесткости и выполнении вычислительных процедур в CAE-программе. Ленточная квадратная матрица имеет следующую структуру:

$$A = \begin{pmatrix} * & * & * & 0 \\ * & \ldots & * & 0 \\ 0 & * & \ldots & * \\ 0 & * & * & * \end{pmatrix}. \tag{7.8}$$

Для ленточных матриц вводят понятие ширины ленты. Для матрицы A с нулевыми элементами $a_{ij} = 0$ при $i > j + m_1$ или $j > i + m_2$ величина $m_1 + m_2 + 1$ называется шириной ленты, а величины m_1 и m_2 называются шириной нижней и верхней полуленты соответственно.

Трехдиагональная матрица – матрица, все элементы которой, кроме элементов главной и примыкающих к ней диагоналей, равны нулю. Важность трехдиагональной матрицы обусловлена тем, что некоторые методы преобразований позволяют привести произвольную матрицу к этому частному виду.

Ортогональная матрица – такая матрица, для которой выполняется условие:

$$A^{-1} = A^T \quad \text{или} \quad A^T A = E\,, \tag{7.9}$$

где A^T – транспонированная матрица, E – единичная матрица. Матрица, обратная ортогональной, эквивалентна транспонированной; определитель ортогональной матрицы равен ± 1.

Сингулярная матрица – такая матрица, между строками которой (а также между столбцами) существует линейная зависимость; определитель такой матрицы равен нулю. Решение системы линейных алгебраических уравнений в CAE-программе приводит к обработке матрицы большой размерности. Если матрица является сингулярной и ее определитель равен нулю, решаемая система уравнений является вырожденной и однозначное решение для нее отсутствует. Понятие вырожденной системы линейных уравнений означает, что фактически сама система уравнений является недостаточно определенной для решения, поскольку некоторые уравнения, входящие в такую систему, представляются линейной комбинацией других уравнений. Тогда существует либо бесконечное множество решений, либо не существует ни одного.

Подобные матрицы – матрицы A и B подобны, если существует такая несингулярная матрица P, что справедливо соотношение

$$B = P^{-1} A P\,. \tag{7.10}$$

Симметрическая матрица совпадает со своей транспонированной матрицей: $A = A^T$, т.е. нижний треугольник квадратной матрицы является «зеркальным отражением» верхнего треугольника. Разреженная ленточная симметрическая матрица может быть объектом обработки в CAE-программах при решении некоторых задач.

Кососимметрическая матрица удовлетворяет условию $A^T = -A$, где A^T – транспонированная матрица.

Симметрическая положительно определенная матрица (в американской литературе – *Symmetric Positive Defined* или сокращенно **SPD**) формируется в CAE-программах при решении задач прочности и жесткости конструкций. Симметрическая матрица $A = A^T$ является положительно определенной $A > 0$, если для любого ненулевого вектора-столбца $\{x\}$ выполняется условие $\{x\}^T [A] \{x\} > 0$. Следствием положительной определенности являются следующие свойства матрицы:

1) все диагональные элементы положительны: $a_{ii} > 0$ для всех i;

2) наибольший по модулю элемент расположен на главной диагонали: $a_{ii} a_{jj} > (a_{ij})^2$ для $i \neq j$.

Треугольная матрица формируется при выполнении вычислительных процедур. Для нижней (*lower*) треугольной матрицы выполняется условие: $a_{ij} = 0$ при $i > j$. Нижняя треугольная матрица имеет структуру:

$$L = \begin{pmatrix} * & 0 & 0 & 0 \\ * & * & 0 & 0 \\ * & * & * & 0 \\ * & * & * & * \end{pmatrix}. \qquad (7.11)$$

Для верхней треугольной матрицы (*upper*) выполняется условие: $a_{ij} = 0$ при $i > j$. Верхняя треугольная матрица имеет структуру:

$$U = \begin{pmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{pmatrix}. \qquad (7.12)$$

Факторизацией матрицы называют такое преобразование, которое ведет к формированию эквивалентной треугольной матрицы. Один шаг факторизации формирует одну строку треугольной матрицы. Такое преобразование производят при решении системы алгебраических уравнений методом исключения (Гаусса), когда коэффициенты системы представлены в матричном виде.

7.4. Прямые методы

Метод Гаусса проводит решение системы уравнений в два этапа. Система уравнений представляется в матричном виде. На первом этапе исходная матрица, соответствующая системе уравнений, с помощью элементарных преобразований (перестановка строк, сложение строк, умножение строки на число) приводится к верхней треугольной матрице. Соответствующая обработка матрицы называется прямым ходом (сверху вниз). После таких преобразований в последней строке остаются нули и значение одной переменной.

Если в процессе приведения матрицы к треугольному виду появляются уравнения, в которых действительное число приравнивается нулю, то это свидетельствует о несовместности системы уравнений. Такая система не имеет ни одного решения.

На втором этапе, т.е. в обратном ходе (снизу вверх), подстановкой находятся последовательно все неизвестные системы уравнений. Источником ошибок в методе Гаусса является умножение строки или столбца матрицы на очень большое число или деление на очень маленькое число. При выполнении таких операций в вычислительных процедурах могут накапливаться большие ошибки, приводящие к появлению значительной погрешности расчетных результатов.

Метод оптимального исключения является модификацией метода Гаусса, требует меньше памяти для решения. Здесь матрица обрабатывается за один проход путем исключения уже выраженных переменных из вышестоящих уравнений.

Метод Крамера преобразует исходную матрицу в произведение двух треугольных матриц, что позволяет свести решение заданной системы к последовательному решению двух систем с треугольными матрицами. Для такого преобразования используются определители матриц. Вычисление определителей матриц высокого порядка осуществляется приближенно и само по себе является трудоемкой операцией, но вычислительная процедура для решения системы уравнений при этом формулируется просто.

Прямые методы имеют ряд недостатков. Как правило, они требуют хранения в оперативной памяти сразу всей матрицы, и при больших значениях n расходуется много места в памяти компьютера. Кроме того, более существенным недостатком прямых методов является накапливание по-

грешностей в процессе решения, поскольку вычисления на любом этапе используют результаты предыдущих операций.

Прямые методы решения линейных систем иногда называют точными, поскольку решение выражается в виде точных формул через коэффициенты системы. Однако точное решение может быть получено лишь при точном выполнении вычислений (и, разумеется, при точных коэффициентах системы). На практике же при использовании компьютеров вычисления проводятся с погрешностями. Поэтому неизбежны погрешности и в окончательных результатах, вызванные погрешностями вычислений (например, погрешностью округления).

Во всех прямых методах накапливается неалгоритмическая вычислительная ошибка. Накопление такой ошибки в вычислениях можно контролировать с помощью так называемой контрольной суммы. Например, в методе Гаусса к каждой строке матрицы добавляют еще один элемент, который равен сумме всех элементов строки (коэффициентов соответствующего уравнения). С этим элементом производят те же операции, что и с коэффициентами уравнения. На каждом шаге проверяют равенство суммы коэффициентов и контрольной суммы: разница говорит о появлении накопившейся вычислительной погрешности. Оценивать можно относительную и абсолютную погрешность. В прямых методах ошибка в вычислениях, когда она не компенсируется случайно другими ошибками, неизбежно ведет к ошибкам в расчетных результатах.

Таким образом, прямыми методами получают решение с погрешностью, появление и нарастание которой можно контролировать. Этой погрешностью в прямых методах трудно управлять, и она может оказаться значительной при высоких порядках системы уравнений или для плохо обусловленных матриц. Плохо обусловленные матрицы характеризуются числом обусловленности. Число обусловленности основной матрицы характеризует уровень погрешности в решении системы уравнений относительно погрешности исходных данных. Если число обусловленности мало, система является хорошо обусловленной. Если число обусловленности велико, то система является плохо обусловленной. Чем меньше число обусловленности, тем меньшей будет погрешность решения относительно погрешностей входных данных.

7.5. Итерационные методы

Из всего многообразия приближенных методов принципиально важными при численном решении различных инженерных задач являются методы итерационные.

Метод итераций (от латинского *iteration* – повторение), строго говоря, следует назвать методом последовательных приближений. Итерационные методы дают возможность найти решение системы как предел бесконечного вычислительного процесса, позволяющего по уже найденным приближениям к решению построить следующее, более точное приближение. Отличительной особенностью итерационных методов является простота вычислительных процедур.

Важное достоинство итерационных методов состоит в том, что погрешности окончательных результатов не накапливаются, поскольку точность вычислений в каждой итерации определяется лишь результатами предыдущей итерации и практически не зависит от ранее выполненных вычислений.

В случае сходящегося итерационного процесса ошибка, полученная в некотором приближении, исправляется в последующих приближениях. Условия и скорость сходимости каждого итерационного процесса существенно зависят от свойств уравнений, т.е. от свойств матрицы системы и от выбора начальных приближений.

Метод простой итерации исходную систему уравнений $[A]\{x\}=\{B\}$ приводит к виду $\{x\}=[C]\{x\}+\{D\}$, где $\det C \neq 0$. За начальное приближение $\{x\}^{(0)}$ решения такой системы уравнений часто берут вектор $\{D\}$:

$$\{x\}^{(0)} = (x_1^{(0)} x_2^{(0)} \ldots x_n^{(0)}) = \{D\} \; . \qquad (7.13)$$

Тогда приближение на первом шаге: $\{x\}^{(1)} = [C]\{x\}^{(0)} + \{D\}$; на втором шаге аналогично: $\{x\}^{(2)} = [C]\{x\}^{(1)} + \{D\}$ или в общем виде

$$\{x\}^{(i)} = [C]\{x\}^{(i-1)} + \{D\} \; . \qquad (7.14)$$

Метод простых итераций можно записать в виде:

$$\begin{aligned}
x_1^{i+1} &= c_{11}x_1^i + c_{12}x_2^i + c_{13}x_3^i + \ldots + c_{1n}x_n^i + d_1 \\
x_2^{i+1} &= c_{21}x_1^i + c_{22}x_2^i + c_{23}x_3^i + \ldots + c_{2n}x_n^i + d_2 \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
x_n^{i+1} &= c_{n1}x_1^i + c_{n2}x_2^i + c_{n3}x_3^i + \ldots + c_{nn}x_n^i + d_n
\end{aligned} \qquad (7.15)$$

Метод Зайделя отличается от метода простых итераций тем, что полученные на текущем шаге значения неизвестных сразу же подставляются в другие уравнения системы. Алгоритм имеет вид:

$$\begin{aligned}
x_1^{i+1} &= c_{11}x_1^i + c_{12}x_2^i + c_{13}x_3^i + \ldots + c_{1n}x_n^i + d_1 \\
x_2^{i+1} &= c_{21}x_1^{i+1} + c_{22}x_2^i + c_{23}x_3^i + \ldots + c_{2n}x_n^i + d_2 \\
x_3^{i+1} &= c_{31}x_1^{i+1} + c_{32}x_2^{i+1} + c_{33}x_3^i + \ldots + c_{3n}x_n^i + d_3 \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
x_n^{i+1} &= c_{n1}x_1^i + c_{n2}x_2^i + c_{n3}x_3^i + \ldots + c_{nn}x_n^i + d_n
\end{aligned} \qquad (7.16)$$

Вопросы для самоконтроля

1. В чем разница прямых и итерационных методов решения систем линейных алгебраических уравнений?
2. В чем заключается прямой и обратный ход метода исключения Гаусса?
3. Что такое прямоугольная, квадратная, транспонированная, блочная, симметрическая, нулевая, единичная, верхняя (нижняя) треугольная, ступенчатая, трехдиагональная, вырожденная и обратная матрицы?
4. При каком условии однородная квадратная система линейных алгебраических уравнений (СЛАУ) имеет ненулевое решение?

ТЕМА 8. Решение дифференциальных уравнений на сетке

«Когда двое делают одно и то же,
получается не одно и то же»
Из законов Мерфи

8.1. Концепция методов решения на расчетной сетке

Многие инженерные задачи приводят к решению дифференциальных уравнений с частными производными, при решении которых нас интересует распределение искомого параметра в пространстве и его изменение во времени в каждой точке исследуемой области пространства. Соответственно искомый параметр является функцией четырех переменных – трех пространственных координат и времени: $f(x,y,z,t)$. Например, если при анализе технической системы ведется поиск температуры, то технического специалиста может интересовать пространственное распределение температуры в анализируемой системе и ее изменение во времени: $T(x,y,z,t)$.

Наиболее важное приложение в технике имеют дифференциальные уравнения первого и второго порядка; напомним, что порядком дифференциального уравнения называется порядок старшей производной. В случае двух независимых переменных x и t дифференциальное уравнение второго порядка с частными производными можно записать в виде

$$F\left(x,t,f,\frac{\partial f}{\partial x},\frac{\partial f}{\partial t},\frac{\partial^2 f}{\partial x^2},\frac{\partial^2 f}{\partial t^2},\frac{\partial^2 f}{\partial x \partial t}\right)=0, \qquad (8.1)$$

где $f=f(x,t)$ – искомая функция, F – заданное выражение, связывающее производные искомой функции.

Решение простейших задач для уравнений с частными производными в ряде случаев может быть проведено аналитически. Это относится к некоторым уравнениям первого порядка, а также к уравнениям второго порядка с постоянными коэффициентами. Однако в подавляющем большинстве случаев при решении научно-технических задач дифференциальные уравнения не могут быть решены аналитически. Для их решения используют приближенные методы, которые требуют выполнения огромного количества элементарных операций и осуществляются с помощью современных

компьютеров, обладающих большим объемом памяти и высокой скоростью вычислений.

Суть приближенных методов решения дифференциальных уравнений в компьютерном инжиниринге заключается в дискретизации задачи: производные в уравнении заменяют приближенными разностными отношениями. Для этого анализируемую область пространства (непрерывную) представляют дискретным множеством точек, которые образуют расчетную сетку и называются расчетными узлами. Исходное дифференциальное уравнение заменяется на сетке системой алгебраических уравнений, решая которую можно найти в узлах сетки приближенные значения искомой функции.

8.2. Метод конечных разностей

В сеточных методах область непрерывного изменения аргумента заменяется дискретным множеством точек, называемых узлами (расчетными узлами, расчетными точками). Множество расчетных узлов составляет разностную сетку (расчетную сетку) (рис. 8.1). Узлы сетки являются расчетными точками, в этих точках вычисляют искомые значения неизвестной переменной.

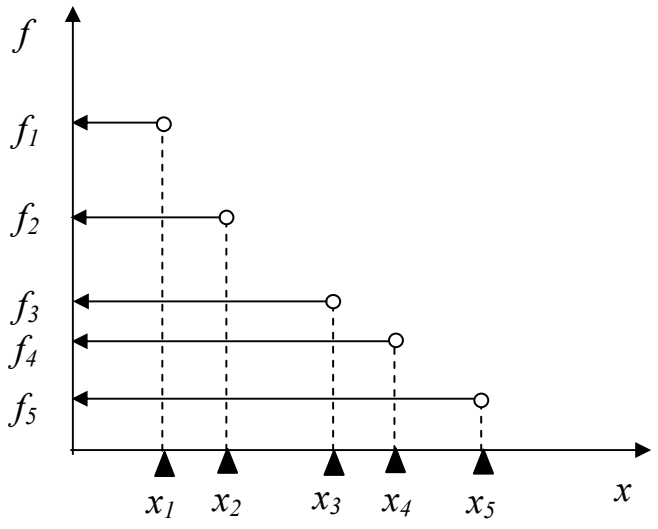


Рис. 8.1. Одномерная разностная сетка: узлы сетки расположены на числовой прямой неравномерно и представлены черными треугольниками

Искомая функция непрерывного аргумента приближенно заменяется функцией дискретного аргумента на заданной сетке. Такая функция называется сеточной.

Производные в дифференциальном уравнении заменяются конечно-разностными соотношениями:

$$\frac{\partial f}{\partial x} \cong \frac{\Delta f_i}{\Delta x_i} \ . \tag{8.2}$$

Замена дифференциального уравнения конечно-разностным производится для всех расчетных узлов и называется аппроксимацией на сетке или разностной аппроксимацией. При этом точные значения искомой функции заменяются значениями сеточной функции в узлах разностной сетки. В итоге формируется система алгебраических уравнений, которая называется разностной схемой. Решая эту систему уравнений, можно найти в узлах сетки значения сеточной функции.

Пример 8.1. Пусть неизвестная величина является функцией двух переменных – времени t и пространственной координаты x. Пусть решаемое дифференциальное уравнение содержит частную производную функции первого порядка по времени и частную производную второго порядка по координате. Необходимо найти распределение неизвестной величины в пространстве и ее изменение во времени. В примере рассматривается одномерная задача, т.е. мы берем во внимание только одну пространственную ось координат. Одномерной моделью можно представить в расчете трубу, рельс, струну, стержень и т.п.

Решение. В такой постановке находят, например, распределение температуры по длине стержня. Зададим по оси стержня расчетные узлы – точки, в которых следует вычислить значения температуры (рис. 8.2). Расстояние h между расчетными узлами зададим, исходя из здравого смысла. Если длина стержня равна 200 мм, то расстояние между расчетными узлами можно задать равным 20 мм. Можно выбрать другое значение, но шаг сетки менее 1 мм для данного одномерного случая не имеет физического смысла. Нет необходимости задать точки с большей плотности, такая дискретность не добавляет принципиально новой информации о распределении температуры в конструкции. Заданы начальные условия, т.е. известны значения температуры в узлах в начальный момент времени. Следует вычислить, как изменяется температурное поле со временем.

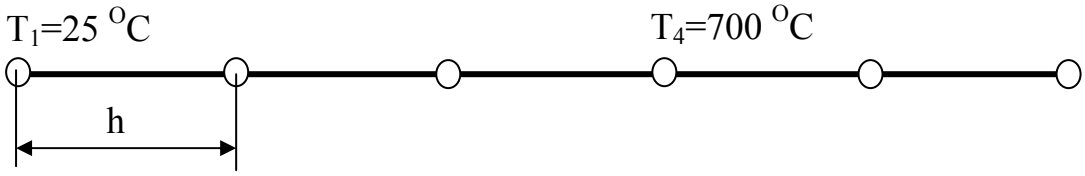


Рис. 8.2. Расчетная сетка и начальные условия

Зададим шаг по времени (рис. 8.3) равным 0.1 секунды.

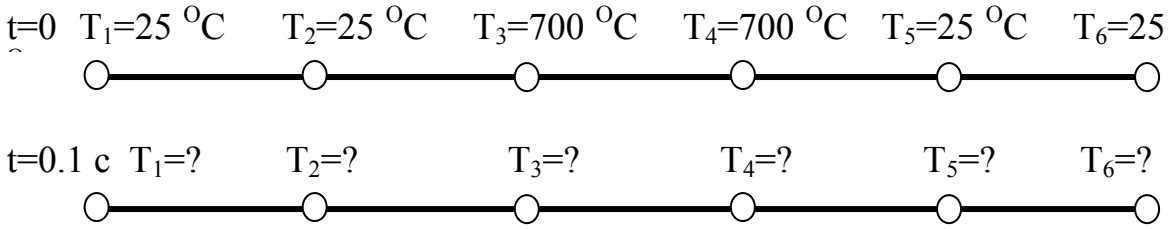


Рис. 8.3. Введение шага по времени с неизвестными значениями температуры в узлах

Изменение температуры в стержне происходит за счет явления теплопроводности. Явление теплопроводности описывается уравнением Фурье, которое содержит первую производную температуры по времени и вторую производную по координате: $\frac{\partial T}{\partial t} = a\frac{\partial^2 T}{\partial x^2}$. Для упрощения записи свойства материала, присутствующие в уравнение теплопроводности, сгруппированы в один коэффициент: $a = \frac{\rho C}{\lambda}$, где ρ – плотность, C – теплоемкость, λ – теплопроводность. Неизвестным параметром, относительно которого решается уравнение теплопроводности, является температура.

Построим разностную схему для уравнения теплопроводности $\frac{\partial T}{\partial t} = a\frac{\partial^2 T}{\partial x^2}$ в случае одномерной передачи тепла в изотропном теле. Искомый параметр – температура $T = T(x,t)$ является функцией двух переменных – пространственной и временной.

При переходе от стержня к набору расчетных точек мы заменили аналитическую задачу дискретной. Соответственно в дифференциальном уравнении производится замена дифференциала константой: $\partial x \approx h$. Помимо пространственного распределения температуры нас также интересует ее изменение в заданных расчетных точках. Выберем шаг расчета по времени и заменим дифференциал константой: $\partial t \approx \tau$.

В итоге, если мы задали, например, 6 точек по длине стержня и 5 расчетных шагов по времени, то нам всего следует определить 30 значений температуры. Разностью значений температуры в двух соседних узлах заменяют дифференциал: $\partial T \approx T_{i+1} - T_i$. Как правило, минимальный набор точек и шагов по времени, отражающий расчетную сетку, группируют в шаблон сетки.

Составим шаблон разностной сетки для уравнения теплопроводности. Пусть индекс i соответствует расстоянию между расчетными узлами в пространстве, индекс j соответствует шагам во времени. В методе конечных разностей шаг сетки фиксируется, величина шага в пространстве равна h, во времени τ. Шаблон разностной сетки представляется графически в виде схемы (рис. 8.4).

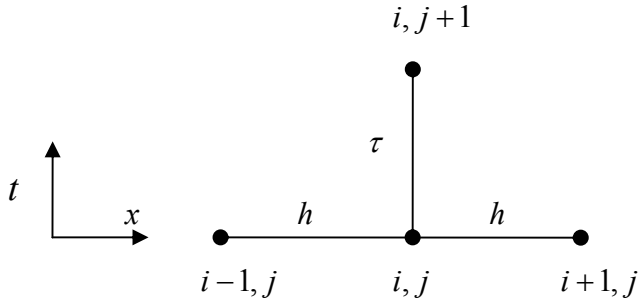


Рис. 8.4. Шаблон разностной сетки для уравнения теплопроводности

Заменяем в дифференциальном уравнении первую и вторую производные конечными разностями значений сеточной функции для точки i, j:

$$\frac{\partial T}{\partial t} \approx \frac{\Delta T}{\Delta t} = \frac{T_i^{j+1} - T_i^j}{\tau} \; ; \tag{8.3}$$

$$\frac{\partial^2 T}{\partial x^2} = \frac{\partial(\frac{\partial T}{\partial x})}{\partial x} \approx \frac{\frac{T_{i+1}^j - T_i^j}{h} - \frac{T_i^j - T_{i-1}^j}{h}}{h} = \frac{T_{i+1}^j - 2T_i^j + T_{i-1}^j}{h^2} \; . \tag{8.4}$$

В каждом узле расчетной сетки заменяем дифференциальное уравнение линейным алгебраическим:

$$\frac{T_i^{j+1} - T_i^j}{\tau} = a \frac{T_{i+1}^j - 2T_i^j + T_{i-1}^j}{h^2} \; . \tag{8.5}$$

□

Здесь использована явная разностная схема, такие схемы применяются для решения уравнений, в которые входят производные по времени. При

замене дифференциального уравнения алгебраическими уравнениями в узлах сетки по явной схеме одно неизвестное значение температуры на следующем шаге вычислений выражается через вычисленные температуры текущего шага. Явные схемы чувствительны к шагу по времени; шаг по времени не может быть произвольным в явных схемах.

8.3. Метод конечных элементов

Решим несколько задач технической механики методом конечных элементов, чтобы показать основную идею этого изящного подхода. Рассмотрим алгоритм формирования матриц в конечно-элементном анализе на простых примерах.

Задача 1. Однокоординатное растяжение упругой пружины

Пружина в конечно-элементной сетке моделируется одномерным упругим линейным элементом с двумя узлами. Задаваемое свойство конечного элемента – жесткость упругой пружины, разрешенное нагружение – узловые силы. Сформируем матрицу жесткости конечного элемента, моделирующего пружину (рис. 8.5) с жесткостью k . Рассматриваемый конечный элемент имеет 2 узла с номерами 1 и 2; действующие в узлах силы f_1 и f_2 вызывают соответствующие перемещения узлов u_1 и u_2.

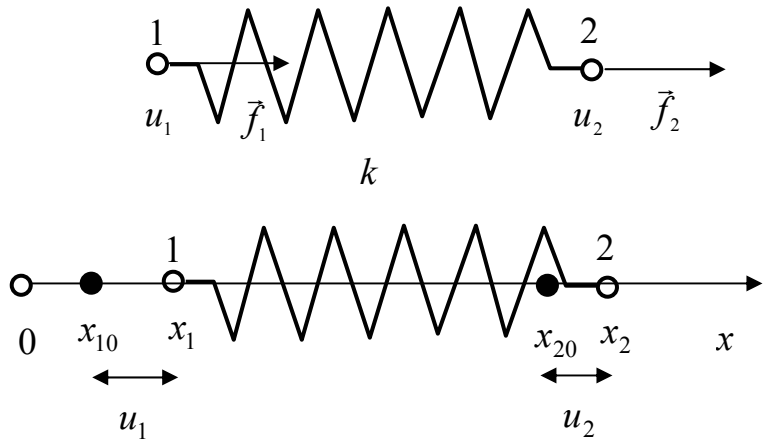


Рис. 8.5. Упругий конечный элемент

Суммарное удлинение конечного элемента, определяющее его деформацию, равно разности перемещений узлов: $(u_2 - u_1)$. Действительно, начальная координата первого узла x_{10}, конечная x_1; начальная координата второго узла x_{20}, конечная x_2. Начальная длина пружины $x_{20} - x_{10}$, конечная длина пружины $x_2 - x_1$. Перемещение первого узла $u_1 = x_1 - x_{10}$, пере-

мещение второго узла $u_2 = x_2 - x_{20}$. Деформация пружины $\varepsilon = (x_2 - x_1) - (x_{20} - x_{10}) = (x_2 - x_{20}) - (x_1 - x_{10}) = u_2 - u_1$.

Сумма сил, приложенных к пружине, равна нулю в статической задаче. Силы, действующие в узлах упругого конечного элемента, подчиняются закону Гука, поэтому связаны с деформациями в узлах линейно через коэффициент пропорциональности, равный жесткости пружины:

$$\begin{cases} f_1 = -k(u_2 - u_1) \\ f_2 = -k(u_1 - u_2) \end{cases} \rightarrow \begin{cases} ku_1 - ku_2 = f_1 \\ ku_2 - ku_1 = f_2 \end{cases} \tag{8.6}$$

или в матричном виде

$$\begin{bmatrix} k & -k \\ -k & k \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix} \tag{8.7}$$

или сжато в матричной форме:

$$[K]\{u\} = \{f\}\ , \tag{8.8}$$

где $[K]$ – матрица жесткости элемента, $\{u\}$ – вектор узловых перемещений, $\{f\}$ – вектор узловых сил.

Обратите внимание на тот факт, что матрица жесткости конечного элемента является симметричной и заполнена с использованием одной величины, описывающей упругие свойства пружины. Коэффициент жесткости можно вынести в виде множителя, отделив квадратную матрицу, наполненную единицами:

$$k\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix}\ .$$

В статической задаче сумма сил, приложенных к конструкции, равна нулю: $f_1 + f_2 = 0$.

Решением задачи являются значения перемещений в узлах. Зная эти перемещения, можно вычислить перемещение любой точки внутри элемента, например, с помощью линейной интерполяционной функции.

$u = ax + b \qquad u_1 = ax_1 + b \quad u_2 = ax_2 + b$

$$u = \frac{u_2 - u_1}{x_2 - x_1} x + \frac{u_2 x_1 - u_1 x_2}{x_2 - x_1}\ .$$

Если поместить начало координат в первый узел и выбрать единичную длину элемента:

$u(x) = (u_2 - u_1)x - u_1$; $u(x) = xu_2 - (x+1)u_1$; $u(x) = \Phi_1(x)u_2 + \Phi_2(x)u_1$.

Это уравнение можно записать в матричной форме

$$u(x) = \{\Phi_1 \quad \Phi_2\}\begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} .$$

Функции $\Phi_1(x)$ и $\Phi_2(x)$ позволяют по известным значениям узловых перемещений вычислить перемещение любой точки внутри элемента. Эти функции являются интерполяционными и называются функциями формы. Они образуют матрицу функций формы $\{\Phi_1 \quad \Phi_2\} \equiv \{-(x+1) \quad x\}$.

Расчетные узлы сетки являются интерполяционными узлами; в них функции формы принимают значения вычисленных ранее узловых перемещений.

Задача 2. Растяжение последовательно соединенных пружин

Усложним задачу – рассмотрим систему, состоящую из двух последовательно соединенных пружин. Представим систему пружин одномерной расчетной сеткой из двух линейных упругих конечных элементов с разной жесткостью (рис. 8.6).

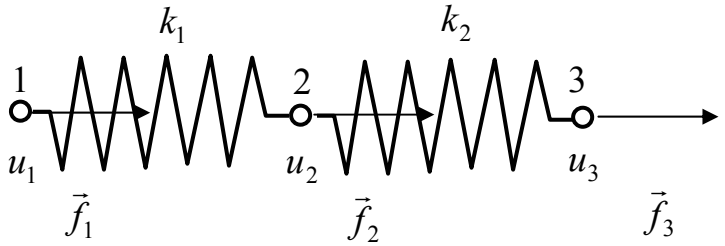


Рис. 8.6. Система из двух упругих элементов

Система уравнений равновесия для первого конечного элемента в матричном виде записывается с использованием матрицы жесткости этого элемента:

$$\begin{bmatrix} k_1 & -k_1 \\ -k_1 & k_1 \end{bmatrix}\begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \begin{Bmatrix} f_1^{\,1} \\ f_2^{\,1} \end{Bmatrix} , \tag{8.9}$$

аналогично для второго конечного элемента:

$$\begin{bmatrix} k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix}\begin{Bmatrix} u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} f_2^{\,2} \\ f_3^{\,2} \end{Bmatrix} , \tag{8.10}$$

где k_1 – жесткость первого элемента,

k_2 – жесткость второго элемента,

$u_1,\ u_2,\ u_3$ – узловые перемещения,

$f_1^1,\ f_2^1$ – узловые силы первого элемента,

$f_2^2,\ f_3^2$ – узловые силы второго элемента.

Суммарные узловые силы $f_1 = f_1^1$, $f_2 = f_2^1 + f_2^2$, $f_3 = f_3^2$ связаны с узловыми перемещениями линейными зависимостями и образуют систему уравнений:

$$\begin{aligned} f_1 &= k_1 u_1 - k_1 u_2 \\ f_2 &= -k_1 u_1 + (k_1 + k_2) u_2 - k_2 u_3 \\ f_3 &= -k_2 u_2 + k_2 u_3 \end{aligned} \qquad (8.11)$$

или в матричном виде:

$$\begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 + k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \\ f_3 \end{Bmatrix}, \qquad (8.12)$$

или кратко в матричном виде:

$$[K]\{u\} = \{f\}, \qquad (8.13)$$

где $[K]$ – глобальная матрица жесткости, составленная из матриц жесткости элементов.

Для пружин с одинаковой жесткостью:

$$k \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \\ f_3 \end{Bmatrix}$$

Замечания по глобальной матрице жесткости

1. Глобальная матрица жесткости является квадратной.

2. Глобальная матрица жесткости является симметричной.

3. Глобальная матрица жесткости конечно-элементной сетки составлена из матриц жесткости конечных элементов.

4. Главные диагонали матриц жесткости конечных элементов совпадают с главной диагональю глобальной матрицы жесткости.

5. На главной диагонали глобальной матрицы жесткости расположены значения жесткости узлов, равные сумме жесткостей примыкающих конечных элементов.

6. Глобальная матрица жесткости содержит нули в отличие от матриц жесткости отдельных конечных элементов.

Замечания по узловым силам

1. В задаче статики сумма сил, приложенных к системе, равна нулю: $f_1 + f_2 + f_3 = 0$.

2. Равны нулю силы в тех узлах, к которым не приложены внешние нагрузки или реакции опор: $f_2 = 0$.

Уточним для данной задачи граничные условия. Поскольку в задачах статики всегда вводятся ограничения степеней свободы, зададим жесткую заделку системы в первом узле, что означает нулевое перемещение узла и действие в нем силы реакции опоры. Приложим к системе внешние силы во втором и третьем узлах, равные F. Граничные условия этой задачи для одномерной конечно-элементной сетки можно записать следующим образом: $u_1 = 0,\ f_2 = f_3 = F$. Тогда система уравнений равновесия имеет вид:

$$\begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 + k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} 0 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} f_1 \\ F \\ F \end{Bmatrix} . \qquad (8.14)$$

Решением системы можно найти неизвестные параметры – силу реакции опоры f_1 и перемещения узлов $u_{2,}\ u_3$ (решением системы из 3 уравнений можно найти 3 неизвестных параметра):

$$f_1 = -2F \ , \quad u_2 = \frac{2F}{k_1} \ , \quad u_3 = \frac{2F}{k_1} + \frac{F}{k_2} \ . \qquad (8.15)$$

Задача 3. Осевое растяжение стержня

Рассмотрим растяжение прямого стержня под действием внешней осевой силы F. Для решения такой задачи следует выбрать одномерный линейный конечный элемент типа «стержень», который позволяет производить расчет конструкции на растяжение-сжатие и кручение.

Исходной геометрической моделью для стержня является отрезок прямой линии длиной a. Переход от одномерной модели к трехмерным расчетам производится за счет реальной константы конечного элемента,

задающей площадь поперечного сечения стержня S. Решение дифференциального уравнения равновесия системы в случае упругой задачи статики требует задания константы, входящей в это уравнение – модуля нормальной упругости материала стержня (модуля Юнга) E.

Сформируем матрицу жесткости конечного элемента, моделирующего стержень (рис. 8.7). Элемент имеет 2 узла с индексами 1 и 2; действующие в узлах силы f_1 и f_2 вызывают соответствующие перемещения узлов u_1 и u_2. Характеристиками элемента являются модуль нормальной упругости E и площадь сечения стержня S.

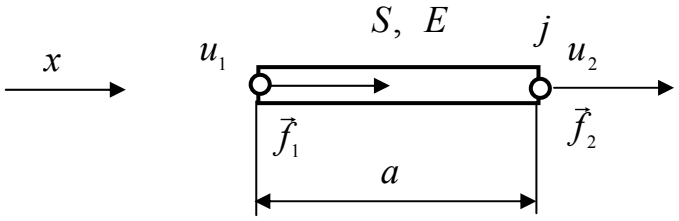


Рис. 8.7. Осевое растяжение стержня

Рассмотрим зависимость осевых перемещений $u = u(x)$, относительных деформаций $\varepsilon = \varepsilon(x)$ и напряжений $\sigma = \sigma(x)$ от координаты точки на оси стержня. В теории упругости устанавливаются известные соотношения между перемещениями, деформациями и напряжениями:

$$\varepsilon = \frac{du}{dx} \quad , \qquad \sigma = E \cdot \varepsilon \quad . \tag{8.16}$$

Перемещение произвольной точки с координатой x внутри элемента можно выразить через узловые перемещения

$$u(x) = u_1 + \frac{x}{a}(u_2 - u_1) = \left(1 - \frac{x}{a}\right)u_1 + \frac{x}{a}u_2 \ , \tag{8.17}$$

тогда выражения (5.16) для деформаций и напряжений преобразуются к виду:

$$\varepsilon = \frac{du}{dx} = \frac{u_2 - u_1}{a} \quad , \qquad \sigma = E \cdot \varepsilon = \frac{E}{a}(u_2 - u_1) \ . \tag{8.18}$$

Запишем очевидное соотношение внешней осевой силы, приложенной к стержню, и перемещений в узлах:

$$F = \sigma \cdot S = \frac{E \cdot S}{a}(u_2 - u_1) = k(u_2 - u_1) \quad , \tag{8.19}$$

где $k = \frac{E \cdot S}{a}$ – жесткость стержня, которая, как видим, зависит от геометрических размеров (длины и площади поперечного сечения стержня) и свойств материала, из которого стержень изготовлен. Стержень, будучи изготовленным из пластика или из алюминия, будет иметь различную жесткость при одних и тех же геометрических размерах.

Матрицу жесткости элемента можно записать следующим образом:

$$[K] = \begin{bmatrix} k & -k \\ -k & k \end{bmatrix} = \begin{bmatrix} \frac{E \cdot S}{a} & -\frac{E \cdot S}{a} \\ -\frac{E \cdot S}{a} & \frac{E \cdot S}{a} \end{bmatrix} = \frac{E \cdot S}{a}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \quad , \tag{8.20}$$

и уравнение равновесия элемента в матричной форме имеет вид

$$\frac{E \cdot S}{a}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \cdot \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix}. \tag{8.21}$$

Обратите внимание на матричное уравнение, которое подлежит решению в конечно-элементном анализе статически нагруженной конструкции. Это уравнение связывает линейной зависимостью узловые перемещения и узловые силы. При решении такого уравнения численными методами находят первичные параметры конструкционного анализа – узловые перемещения. Вычисление узловых перемещений не требует задания предельных характеристик конструкционного материала – предела прочности или предела текучести, эти константы не входят в решаемое уравнение.

Задача 4. Осевое растяжение ступенчатого стержня

Рассмотрим решение на конечно-элементной сетке классической задачи из курса технической механики об осевом растяжении ступенчатого стержня. В задаче ступенчатый стержень (рис. 8.8) с двумя ступенями одинаковой длины a и площадью поперечного сечения ступеней соответственно S_1 и S_2 жестко заделан и нагружен осевой силой F. Следует вычислить перемещения сечений 1, 2 и 3.

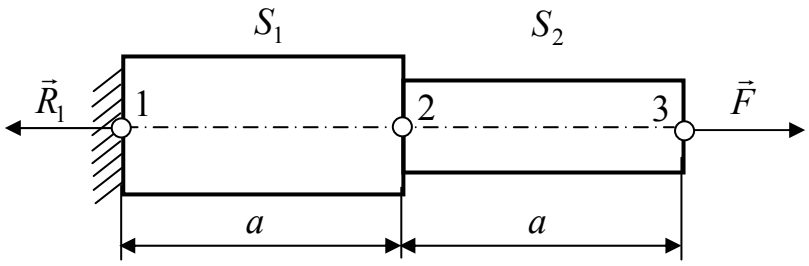


Рис. 8.8. Осевое растяжение ступенчатого стержня

Представим стержень сеткой из двух стержневых элементов (1 и 2) и трех узлов (1, 2 и 3). Перемещение сечений равны перемещениям соответствующих узлов сетки u_1, u_2, u_3. Рассмотрим элемент 1 (рис. 8.9) длиной a и поперечным сечением S_1, в узлах элемента приложены усилия f_1 и f_2, которые вызывают соответствующие перемещения этих узлов u_1 и u_2.

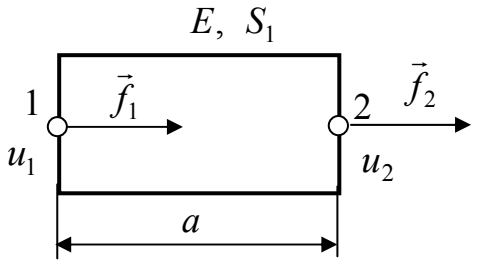


Рис. 8.9. Перемещения в конечном элементе

Запишем очевидные соотношения сил и перемещений в узлах:

$$f_1 = \frac{E \cdot S_1}{a}(u_1 - u_2) \quad , \qquad f_2 = \frac{E \cdot S_1}{a}(u_2 - u_1) \tag{8.22}$$

или обобщенно в матричной форме

$$\frac{E \cdot S_1}{a}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix} \tag{8.23}$$

или кратко в матричной форме

$$[K]\{u\} = \{f\} \ , \tag{8.24}$$

где E – модуль нормальной упругости (модуль Юнга) материала стержня, $\frac{E \cdot S_1}{a}$ – жесткость элемента,

$[K]$ – матрица жесткости элемента, связывающая линейно узловые усилия и узловые перемещения.

Составим далее глобальную матрицу жесткости для всей сеточной модели из матриц жесткости отдельных элементов. При этом главные диагонали матриц жесткости элементов совпадают с главной диагональю глобальной матрицы жесткости и стыкуются в узле 2. Общую систему уравнений равновесия для сетки можно записать развернуто:

$$\frac{E}{a}\begin{bmatrix} S_1 & -S_1 & 0 \\ -S_1 & S_1+S_2 & -S_2 \\ 0 & -S_2 & S_2 \end{bmatrix}\begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} -R_1 \\ 0 \\ F \end{Bmatrix}, \qquad (8.25)$$

где R_1 – реакция опоры в узле 1,

F – осевая сила, приложенная в узле 3.

Как видим, узел 2 находится во внутренней области сетки, к нему не прикладываются граничные условия, т. е. этот узел свободен от внешних нагрузок.

Учтем, что $k_1 = \frac{E \cdot S_1}{a}$ и $k_2 = \frac{E \cdot S_2}{a}$ – значения жесткости для первого и второго элементов соответственно, тогда глобальную матрицу жесткости можно выразить через значения жесткости элементов:

$$\begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1+k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix}\begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} -R_1 \\ 0 \\ F \end{Bmatrix}. \qquad (8.26)$$

Обратите внимание на тот факт, что конструкция выполнена из одного материала с заданным модулем нормальной упругости E, и жесткость элемента k зависит только от геометрических размеров – длины и площади сечения соответствующей ступени стержня.

Добавим граничные условия в виде нулевого перемещения в точке заделки (ограничение степеней свободы в узле): $u_1 = 0$:

$$\begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1+k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix}\begin{Bmatrix} 0 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} -R_1 \\ 0 \\ F \end{Bmatrix}. \qquad (8.27)$$

Решение этой системы алгебраических уравнений:

$$u_1 = 0\ , \quad u_2 = \frac{Fa}{ES_1}\ , \quad u_3 = \frac{Fa}{E}\left(\frac{1}{S_1}+\frac{1}{S_2}\right). \qquad (8.28)$$

Вопросы для самоконтроля

1. Постройте шаблон разностной сетки и замените волновое уравнение $\frac{\partial^2 u}{\partial t^2} = a\frac{\partial^2 u}{\partial x^2}$ разностным.
2. Каким методом можно найти значение искомого параметра в точке между узлами сетки?

ТЕМА 9. Оптимизация

«Усложнять – просто, упрощать – сложно»
Из законов Мерфи

9.1. Концепция методов оптимизации

Многие технические проблемы приводят инженеров к решению задач оптимального проектирования, т.е. к поиску наилучшего варианта технологии или конструкции. Такие задачи решаются с использованием различных методов оптимизации, которые основаны на поиске экстремума некоторой функции, отражающей критерий оптимальности, при условии выполнения ряда ограничений.

Оптимизация – процесс выбора наилучшего варианта технической системы из всех возможных. Оптимизация направлена на поиск тех важных параметров, от которых зависит реализация наилучшего состояния технической системы. Такие параметры называются проектными параметрами. В процессе решения задачи оптимизации необходимо найти оптимальные значения проектных параметров, определяющих наилучшее состояние технической системы. Проектными параметрами в задачах конструкционной прочности могут быть, например, линейные размеры конструкции, масса или температура. Количество искомых проектных параметров n характеризует размерность задачи оптимизации.

Выбор оптимального решения или сравнение нескольких альтернативных вариантов технической системы проводится с помощью некоторой зависимой величины, определяемой проектными параметрами. Такая величина является функцией проектных параметров и называется целевой функцией (другие названия – целевой параметр, критерий оптимальности, критерий качества).

В процессе решения задачи оптимизации должны быть найдены такие значения проектных параметров, при которых целевая функция достигает минимума (или максимума). Целевыми параметрами в задачах конструкционной прочности могут быть, например, прочность, масса, мощность, объем. Целевых функций может быть несколько. Причем целевые функции могут оказаться несовместимыми и противоположным образом влиять на проектные параметры. Например, при проектировании детали необходимо одновременно обеспечить максимальную прочность и минимальную материалоемкость. В таких случаях вводится приоритет целевых функций. В первую очередь при проектировании детали обеспечивается ее прочность, а затем берется во внимание материалоемкость.

Методы оптимизации можно классифицировать 1) по размерности решаемой задачи (одномерные и многомерные); 2) по наличию ограничений (условные и безусловные); 3) по способу формирования шага (градиентные, безградиентные, случайный поиск).

Пример 9.1. Требуется спроектировать закрытый сварной контейнер из металлического листа в форме прямоугольного параллелепипеда объемом $V = 1$ м^3 и израсходовать на его изготовление как можно меньше металла.

При постоянной толщине стенок сформулированное в техническом задании требование означает, что площадь полной поверхности контейнера S должна быть минимальной. Обозначим x_1, x_2, x_3 – длины ребер контейнера. Минимизируемая площадь является функцией линейных размеров контейнера и имеет вид $S = 2(x_1x_2 + x_2x_3 + x_1x_3)$. Здесь x_1, x_2, x_3 – проектные параметры, S – целевая функция, $V = 1$ – ограничение равенства, которое позволяет исключить один параметр из списка искомых проектных параметров и таким образом снизить размерность задачи:

$$V = x_1x_2x_3 = 1 \qquad \rightarrow x_3 = \frac{1}{x_1x_2} \ .$$

С учетом данного ограничения задача сводится к минимизации целевой функции двух переменных:

$$S = 2(x_1x_2 + \frac{1}{x_1} + \frac{1}{x_2}) \rightarrow \min \ .$$

□

9.2. Одномерная оптимизация

Задачу оптимизации можно сформулировать таким образом, что основной вычислительной процедурой становится поиск минимума целевой функции $R(x)$. В связи с этим представляют интерес численные методы поиска экстремумов целевой функции. Пусть целевая функция является функцией одного искомого проектного параметра $R(x)$. Подобные целевые функции формируются в задачах одномерной оптимизации.

Рассмотрим одномерные задачи оптимизации вида

$$R(x) \to \min[a \le x \le b] \,, \tag{9.1}$$

где скалярный проектный параметр x задан на отрезке $[a,b]$.

Экстремум целевой функции можно найти построением улучшающей последовательности значений проектного параметра x_0, x_1, ..., x_i, x_{i+1}, где $\left|x_i - x_{i+1}\right| \le \varepsilon$, ε – задаваемая погрешность решения. Поиск экстремума в этом случае ведется пошагово: на каждом шаге вычисляется следующее значение последовательности. Вычисление останавливается, когда модуль разности двух последовательно вычисленных значений проектного параметра оказывается меньше объявленной погрешности. Последнее вычисленное значение проектного параметра является решением задачи оптимизации с заданной точностью.

Метод сканирования заключается в последовательном переборе всех значений $a \le x \le b$ с шагом, равным погрешности решения ε, или с переменным шагом, величина которого назначается путем последовательного уточнения решения. На каждом шаге вычисляется целевая функция R, затем выбирается наименьшее из всех вычисленных значений R и назначается соответствующее минимуму целевой функции решение x.

Более быстрый алгоритм поиска экстремума последовательным уточнением включает на первом этапе сканирование с крупным шагом Δ_1. Затем отрезок, внутри которого получено наименьшее значение целевой функции, разбивается на более мелкие отрезки Δ_2 и т.д. (рис. 9.1) до тех пор, пока величина отрезка не окажется меньше заданной погрешности. Достоинство метода в том, что можно найти глобальный минимум целевой функции, когда существует несколько экстремумов.

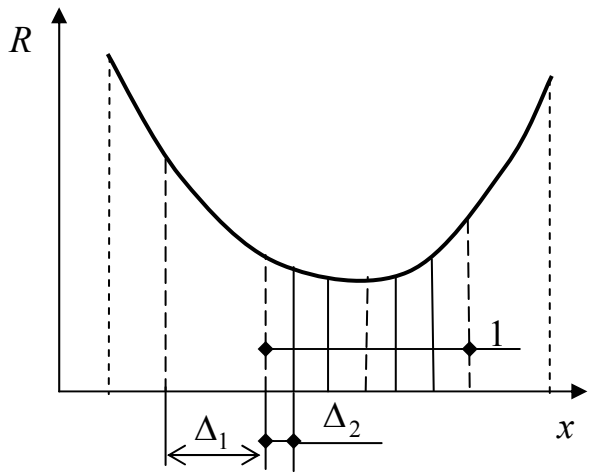


Рис. 9.1. Поиск экстремума методом сканирования

Пример 9.2. Требуется найти экстремум целевой функции $R(x) = 2\sin(x+1)$ на отрезке $[1 \le x \le 2]$. Ошибка задается по переменной x: $\varepsilon = 0.05$.

□

Метод деления пополам основан на делении текущего отрезка $[a,b]$, где содержится искомый экстремум, на две равные части. В качестве следующего отрезка выбирается та половина, в которой локализуется экстремум. Экстремум локализуется путем сравнения двух значений целевой функции в точках, отстоящих от середины текущего отрезка на $\varepsilon/2$, где ε – погрешность решения задачи оптимизации. Если $R(x+\varepsilon/2) < R(x-\varepsilon/2)$, то минимум располагается в правой половине текущего отрезка $[a,b]$, в противном случае – в левой. Процесс поиска завершается, когда величина текущего отрезка становится меньше заданной погрешности.

9.3. Градиентная оптимизация

При градиентной оптимизации величина шага в рекуррентном соотношении вычисляется с использованием градиента целевой функции. Направление градиента показывает направление наискорейшего убывания функции, а его модуль – скорость этого убывания.

Метод градиента формирует шаг $h \cdot grad\ R(x)$ по переменной x как функцию от градиента целевой функции в текущей точке поиска; причем коэффициент пропорциональности h управляет эффективностью поиска; снижение h вблизи оптимального решения, например, позволяет быстрее найти решение.

Простейший алгоритм поиска минимума целевой функции записывается в векторной форме следующим образом:

$$x^{i+1} = x^i - h \ \ grad \ \ R(x^i) \tag{9.2}$$

или в скалярном виде для целевой функции нескольких проектных параметров:

$$x_j^{i+1} = x_j^i - h \frac{\partial R}{\partial x_j^i} \ , \ j = 1,...,n \ . \tag{9.3}$$

Поиск каждой новой точки на пути к оптимальному решению состоит из двух этапов: 1) оценка градиента целевой функции $R(x)$ путем вычисления ее частных производных по каждому проектному параметру x_j; 2) осуществление рабочего шага по всем переменным одновременно.

Сравнительная иллюстрация траекторий, по которым градиентные методы находят минимум целевой функции, приведена на рис. 9.2.

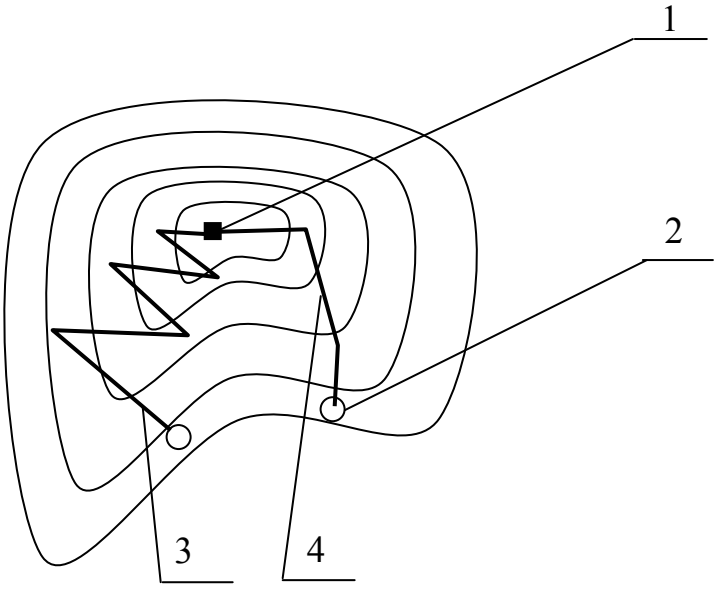


Рис. 9.2. Сравнительная иллюстрация траекторий поиска минимума двумерной целевой функции градиентными методами:
1 – оптимум, 2 – начальные точки,
3 – метод градиента, 4 – метод сопряженных градиентов

Метод наискорейшего спуска. Основным недостатком градиентного метода является необходимость частого вычисления производных от целевой функции $R(x)$. Этого недостатка лишен метод наискорейшего спуска, где в текущей точке вычисляется $grad \ \ R(x)$, а затем в направлении гради-

ента вычисляется минимум целевой функции $\min\ R(x)$. Практически это может быть осуществлено любым методом одномерной оптимизации как поиск по одному направлению – направлению градиента; наиболее часто используется сканирование до первого локального минимума по направлению $grad\ R(x)$.

Вдали от оптимума эффективность метода высока, что позволяет быстро выйти в район оптимума; но в окрестности оптимума эффективность метода снижается из-за частой смены направления поиска. Метод, как и все градиентные методы, обладает невысокой эффективностью в так называемых «овражных» функциях, когда в области оптимума существует несколько локальных экстремумов. В ряде случаев можно повысить скорость выхода в район оптимума предъявлением невысоких требований к точности поиска минимума целевой функции по направлению градиента, который регулируется коэффициентом h.

Метод сопряженных градиентов. На начальных этапах поиска решения метод сопряженных градиентов работает так же, как метод наискорейшего спуска. В пределах шага нелинейная целевая функция заменяется линейным полиномом и производится вычисление градиента целевой функции через первые производные по координатам. Поскольку для приближения целевой функции используется в таких вычислениях полином первого порядка, градиентный метод наискорейшего спуска относят к методам первого порядка.

Метод сопряженных градиентов в область оптимума выходит так же, как метод первого порядка, а уже в окрестностях оптимума трансформируется в метод второго порядка и анализирует для текущей точки вторые производные от целевой функции.

При переходе к численному анализу вторых производных алгоритм метода сопряженных градиентов выбирает направление поиска в виде линейной комбинации векторов градиента в данной точке и предшествующего направления:

$$\begin{aligned} x^1 &= x^0 - h\ grad\ R(x^0), \\ x^{i+1} &= x^i - h\ [grad\ R(x^i) + \alpha\ grad\ R(x^{i-1})]\ , \end{aligned} \tag{9.4}$$

где коэффициент α вычисляется как квадрат отношения длин векторов градиентов на текущем шаге и на предыдущем шаге:

$$\alpha = \frac{\left|grad\ R(x^i)\right|^2}{\left|grad\ R(x^{i-1})\right|^2}\ . \qquad (9.5)$$

Вопросы для самоконтроля

1. Экстремум каких целевых функций $R(x)$ можно найти методом сканирования?
2. Как повысить точность поиска оптимального решения?
3. Что называется градиентом целевой функции $R(x)$?
4. Найти оптимальные размеры контейнера в примере 9.1.

ТЕМА 10. Собственные значения и векторы

«Оставшиеся гайки никогда
не подходят к оставшимся болтам»
Из законов Мерфи

10.1. Понятие собственных значений

Применение компьютерных систем инженерного анализа к расчету конструкций и механических систем предполагает численное решение задач механики, в ходе которого возникает необходимость решать большие системы алгебраических уравнений. Некоторые задачи механики в такой постановке требуют нахождения собственных или характеристических значений системы.

С задачами на определение собственных значений и собственных векторов инженер сталкивается при проектировании различных систем и сооружений. Так, при анализе напряженного состояния конструкции для тензоров напряжений собственные значения определяют главные нормальные напряжения, а собственными векторами задаются направления, связанные с тройкой значений главных напряжений.

При динамическом анализе механических систем, например, при модальном анализе вибраций собственные значения соответствуют собственным частотам колебаний, а собственные векторы характеризуют соответствующие деформации конструкции, соответствующую каждой собственной частоте колебаний. При расчете конструкций на устойчивость собственные

значения позволяют определять критические нагрузки, превышение которых приводит к потере устойчивости.

В общем виде задача на собственные значения квадратной матрицы формулируется следующим образом:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{n2} \end{bmatrix} \cdot \begin{Bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{Bmatrix} = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix} \cdot \begin{Bmatrix} x_1^{\,0} \\ x_2^{\,0} \\ \dots \\ x_n^{\,0} \end{Bmatrix} \qquad (10.1)$$

или кратко: $[A] \cdot \{x\} = [\lambda] \cdot \{x^0\}$

где A – квадратная матрица размерности $n \times n$.

Квадратная матрица A размерности $n \times n$ имеет n собственных значений λ_i ; набор всех ее собственных значений называется спектром матрицы. Нумерацию собственных значений производят по возрастанию модуля:

$$|\lambda_1| \le |\lambda_2| \le \dots \le |\lambda_{n-1}| \le |\lambda_n| \ . \qquad (10.2)$$

Каждому собственному значению λ_i соответствует собственный вектор x_i . В общем случае требуется найти n собственных скалярных значений λ и собственные векторы x^0, соответствующие каждому из собственных значений.

Собственные векторы x_i^0 образуют систему ортонормированных векторов:

$$\{x_i^{\,0}\} \cdot \{x_k^{\,0}\} = \begin{cases} 1, & i = k \\ 0, & i \ne k \end{cases} \ . \qquad (10.3)$$

Привлечение компьютерных методов к расчетному обоснованию проектируемых инженерных систем сводит решение исходного дифференциального уравнения, описывающего физические процессы в системе, к обработке квадратной матрицы большой размерности. Соответственно, рассматривая компьютерные методы решения инженерных задач, следует обсудить методы поиска собственных значений и векторов квадратной матрицы.

10.2. Концепция поиска собственных значений

Алгоритмы решения задач на собственные значения квадратной матрицы можно разделить на две группы. 1) Существуют итерационные методы, которые удобны и хорошо приспособлены для определения наименьшего и наибольшего собственных значений. 2) Существуют также методы преобразований подобия, которые сложней, но зато позволяют определить все собственные значения и собственные векторы матрицы.

Наиболее очевидным способом решения задачи на собственные значения является их определение из системы уравнений:

$$([A]-[\lambda]\cdot[E])\cdot\{x\}=0, \qquad (10.4)$$

которая имеет ненулевое решение лишь в случае, если det $(A - \lambda E) = 0$.

Уравнение det $(A - \lambda E) = 0$ является алгебраическим уравнением, его называют характеристическим уравнением. Например, для квадратной матрицы второго порядка характеристическое уравнение имеет вид

$$\det([A]-\lambda\cdot[E])=\det\begin{bmatrix}a_{11}-\lambda & a_{12}\\ a_{21} & a_{22}-\lambda\end{bmatrix}=(a_{11}-\lambda)(a_{22}-\lambda)-a_{12}a_{21}=0\ .$$

Раскрыв определитель, получим многочлен n-й степени относительно λ, корни которого и будут собственными значениями матрицы. Итак, собственные значения квадратной матрицы A определяются как корни характеристического полинома det $(A - \lambda E)$. Для определения корней можно воспользоваться любым из известных численных методов.

Вопросы для самоконтроля

1. Какая матрица называется квадратной?
2. Что будет результатом умножения квадратной матрицы на единичную?
3. Как вычисляют определитель матрицы?

ТЕМА 11. Параллельные вычислительные технологии

«Лишь в конце работы мы обычно понимаем,
с чего ее нужно было начинать»
Из законов Мерфи

Рассматривая параллельные вычислительные технологии, следует разделить понятия «параллельные вычислительные системы» и «параллельные численные методы».

11.1. Параллельные вычислительные системы

Характерной особенностью современных вычислительных систем является возможность одновременного или, как говорят, параллельного использования большого числа процессоров для обработки информации. Для таких систем собирательным стало название «параллельные вычислительные системы». Необходимость эффективного их применения требует радикального изменения структуры численных методов. Разнообразие архитектур вычислительных систем порождает разнообразие способов организации вычислений, что вызывает различие в организации данных, численных методов и алгоритмов, а также вариативность средств и языков общения с вычислительной техникой. Для больших параллельных систем разрабатываются специальные численные методы и пишутся прикладные программы с учетом архитектуры таких систем.

Проблема согласования программных средств и структуры вычислительной техники появилась в связи с возможностью выполнения вычислений на многопроцессорных комплексах, поскольку традиционные алгоритмические языки достаточно хорошо согласованы с классическими однопроцессорными машинами. Внешнее совпадение операций алгоритмических языков с математическими операциями создает устойчивую иллюзию совпадения их свойств и скрывает от пользователя большинство проблем реализации алгоритмов на конкретных вычислительных устройствах. Однако традиционное машинно-независимое программное обеспечение оказывается неэффективным в случае параллельных вычислительных систем. Для этих систем становится актуальной проблема согласованности алгоритмов и алгоритмических языков со структурой вычислительной системы. Эффективной реализации алгоритма могут помешать, например, особенности коммуникационной сети, связывающей процессоры между собой и с

ячейками памяти. Ограниченные возможности быстрых связей коммуникационной сети могут быть не согласованными со структурой связей в алгоритме.

Вычислительная система представляет совокупность связанных между собой устройств. В каждый момент времени эти устройства либо простаивают, либо выполняют полезную работу, т.е. заняты хранением, пересылкой или переработкой информации. Системы различаются как составом устройств, так и видом связей. Связи могут быть постоянными или изменяемыми.

Основные структурные элементы вычислительной системы можно ввести и описать на примере однопроцессорной машины (рис. 11.1).

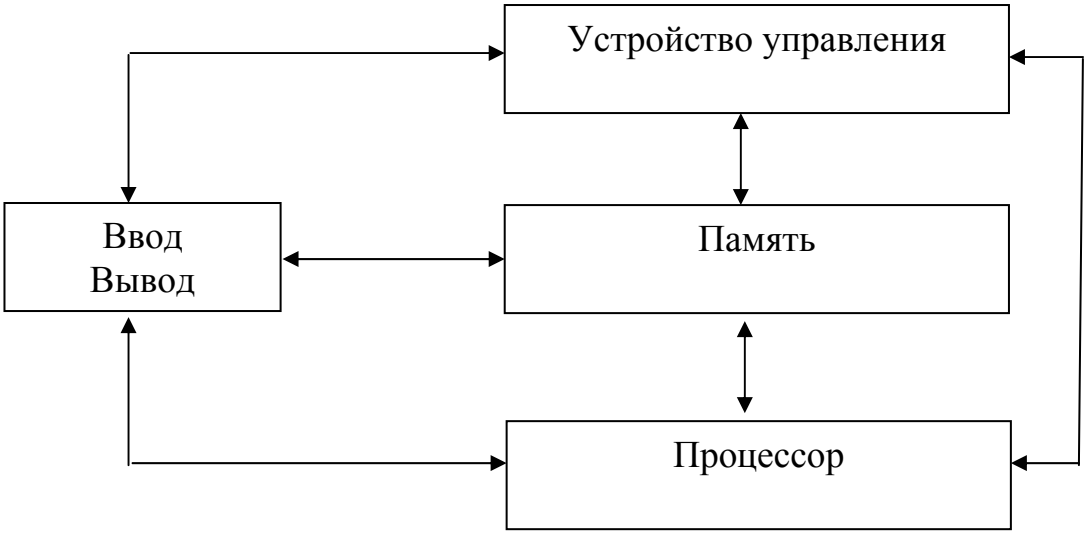


Рис. 11.1. Структура однопроцессорной машины

В однопроцессорной машине имеется два основных устройства. Одно из них называется процессором (центральный процессор, решающее устройство, арифметико-логическое устройство) и предназначено для выполнения над операндами некоторого ограниченного набора операций. В набор операций входят операции сложения, вычитания, умножения и деления чисел, логические операции над отдельными разрядами и их последовательностями, операции над символами и др. Другое устройство, называемое памятью (запоминающее устройство), предназначено для хранения всей информации, необходимой для организации работы процессора. Процессор является активным устройством, т.е. он имеет возможность преобразовывать информацию. Память является пассивным устройством, т.е. она не имеет такой возможности. Процессор и память связаны между собой каналами обмена информацией.

Работа однопроцессорной машины заключается в последовательном выполнении отдельных команд. Каждая команда содержит информацию о том, какая операция из заданного набора должна быть выполнена, а также из каких ячеек памяти должны быть взяты аргументы операции и в какие ячейки помещен результат. Описание упорядоченной последовательности команд в виде программы находится в памяти. Там же размещаются необходимые для реализации алгоритма начальные данные и результаты промежуточных вычислений. Координирует работу всех узлов машины устройство управления. Оно организует последовательную выборку команд из памяти и их расшифровку, передачу из памяти в процессор операндов, а из процессора в память результатов выполнения команд, управляет работой процессора. Ввод начальных данных и выдачу результатов осуществляет устройство ввода-вывода.

Процессор является единственным устройством, которое выполняет полезную с точки зрения реализации алгоритма работу. Все остальные устройства по отношению к нему оказываются обслуживающими, и их работа направлена только на то, чтобы обеспечить наиболее эффективный режим функционирования процессора. В основе архитектуры описанной машины лежит принцип последовательного выполнения отдельных команд. Организованные таким образом машины называют последовательными.

Повысить быстродействие при решении больших задач можно, объединив несколько процессоров и несколько устройств памяти в единую вычислительную систему с помощью каналов передачи информации. Коммуникационная сеть каналов связи между устройствами лимитирует быстродействие такой системы, поскольку скорость обработки информации процессорами существенно превышает скорость передачи данных между устройствами. Причем с ростом миниатюризации элементной базы и устройств вычислительной техники проблема построения коммуникационных сетей не только не снимается, но становится еще более сложной и актуальной. Прямая связь каждого устройства со всеми остальными приводит к резкому усложнению сети с ростом числа устройств. Каждый процессор параллельной системы может иметь свою собственную память, и тогда память системы называется распределенной. Для распределенной памяти характерно наличие большого числа быстрых каналов, связывающих отдельные ее части с отдельными процессорами. Обмен информацией между частями распределенной памяти обычно осуществляется довольно медленно.

Если процессоры общаются с памятью через общие каналы связи, то в этом случае память называется общей. Для параллельных систем с общей памятью одним из узких мест, затрудняющих достижение максимального быстродействия, являются каналы связи с памятью. Одна и та же вычислительная система может иметь как общую, так и распределенную память.

Стремление эффективно решать задачи на многопроцессорных системах должно обязательно сопровождаться согласованием структуры численных методов и архитектуры вычислительных систем. Сопряжение вычислительных систем и численных методов порождает два класса проблем – обеспечение быстрой коммуникации и полная загрузка процессоров. Наращивание мощности вычислительных систем с многими процессорами и распределенной памятью в какой-то момент приводит к тому, что возможности коммуникационной сети в обеспечении быстрой передачи данных достигают предела и начинают ограничивать быстродействие системы в целом. Причина заключается в том, что стремление увеличить быстродействие за счет использования большого числа процессоров может быть несовместимым в условиях применения конкретного численного метода с возможностями коммуникационной сети. В итоге любая вычислительная система с большим количеством процессоров будет эффективной только на определенном классе алгоритмов и методов.

Не только ограничения коммуникационной сети сдерживают рост производительности вычислительной системы. Требуется также постоянная загруженность всех процессоров выполнением полезной работы. Но алгоритм решения задачи в силу причин, вызванных его собственной структурой, может просто не обладать возможностью обеспечить постоянную загрузку большого числа процессоров независимо от устройства коммуникационной сети. Имеется много численных методов, которые нельзя эффективно реализовать ни на каких многопроцессорных системах.

Классификация параллельных вычислительных систем учитывает следующие признаки: тип потока данных, тип потока команд, способ обработки данных, тип строения памяти, тип коммуникационной сети, степень однородности компонент системы, степень согласованности режимов работы устройств. Широко известно деление систем по двум признакам: тип потока команд между устройствами памяти и устройствами управления или процессорами и тип потока данных между процессорами и устройствами памяти. Поток команд может быть одиночным (*Single Instruction*) или

множественным (*Multiple Instruction*). В одиночном потоке в один момент времени может выполняться только одна команда, и эта единственная команда определяет работу многих устройств в данный момент. В множественном потоке в один момент времени может выполняться много команд, и каждая команда определяет работу только одного устройства в данный момент. В одиночном потоке последовательно выполняются отдельные команды, в множественном потоке – группы команд.

В свою очередь, одиночный поток данных (*Single Data*) предполагает использование только одного устройства памяти и одного процессора. При этом используемый процессор может быть простым или очень сложным, так что процесс обработки единицы информации потока будет требовать выполнения многих команд. Множественный поток данных (*Multiple Data*) состоит из многих одиночных потоков, полностью независимых или связанных использованием общей памяти или общего процессора.

В соответствии с данной классификацией все параллельные системы делятся на четыре класса: **SISD**, **SIMD**, **MISD**, **MIMD**. Однопроцессорная машина классической структуры является типичным представителем класса SISD. К классу MISD можно отнести систему, имеющую сложный процессор с выполнением отдельных операций (сложение, умножение, сдвиг, логическое умножение и т.п.) специализированными устройствами последовательно по конвейерному типу. При работе конвейера положительный эффект достигается за счет выполнения большого числа независимых операций: для того чтобы начать выполнение следующей операции, не нужно ждать окончания всего процесса выполнения предыдущей операции; достаточно, чтобы у предыдущей операции был закончен только первый этап. Реализация конвейера позволяет максимально загрузить процессоры выполнением полезной работы. Однако за достижение этого эффекта приходится платить необходимостью организовывать данные особым образом, объединяя в отдельные потоки независимые однотипные данные. Обратите внимание, что и в данном случае эффективность процесса реализации задачи решающим образом зависит от степени согласованности структуры численных методов и особенностей архитектуры вычислительных систем.

Системы класса SIMD и MIMD всегда имеют много процессоров, работающих параллельно. В системах MIMD процессоры, как правило, универсальные, имеют собственную память и редко обмениваются информацией между собой. Коммуникационная сеть таких систем обеспечивает бы-

стрые передачи информации только между процессорами и их собственной памятью; остальные обмены осуществляются значительно медленнее. В системах SIMD процессоры могут быть как универсальными, так и специализированными. Обычно они имеют небольшую собственную память или не имеют ее совсем. Коммуникационная сеть системы SIMD обеспечивает быстрые передачи информации как между процессорами и их собственной памятью, так и между некоторыми из процессоров.

11.2. Параллельные алгоритмы

Как бы ни были устроены параллельные вычислительные системы, все они обладают одной общей особенностью: в каждый момент времени преобразование информации может осуществляться одновременно на многих функциональных устройствах, причем на каждом устройстве информация в данный момент времени преобразуется независимо от остальных. Соответственно, алгоритм, реализуемый на параллельной системе, представляется в виде последовательности групп операций. Все операции одной группы должны быть независимыми и обладать возможностью быть выполненными одновременно на имеющихся в системе функциональных устройствах.

Пусть операции алгоритма разбиты на группы, упорядоченные так, что каждая операция любой группы зависит либо от начальных данных алгоритма, либо от результатов выполнения операций, находящихся в предыдущих группах. Представление алгоритма в подобном виде называется параллельной формой алгоритма. Каждая группа операций называется ярусом параллельной формы, число групп – высотой параллельной формы, максимальное число операций в ярусе – шириной параллельной формы.

Если известна параллельная форма алгоритма, то сам алгоритм можно реализовать на параллельной вычислительной системе по шагам последовательно ярус за ярусом. Если состав функциональных устройств системы таков, что параллельно могут выполняться все операции любого яруса, то при подходящим образом организованной коммуникационной сети алгоритм может быть реализован на параллельной системе за время, пропорциональное высоте параллельной формы.

Разрабатываемый алгоритм может иметь много параллельных форм, различающихся высотой и шириной. Но среди всех параллельных форм для данного алгоритма имеется одна форма минимальной высоты. Такая па-

раллельная форма с минимальной высотой называется максимальной, а ее высота определяет высоту алгоритма. Максимальная параллельная форма алгоритма представляет наибольший интерес, поскольку она определяет минимальное время реализации алгоритма на параллельной вычислительной системе.

Пример 11.1. Составить параллельные формы для алгоритма вычислений согласно алгебраическому выражению $(a_1a_2 + a_3a_4)(a_5a_6 + a_7a_8)$.

Решение. Для данного алгоритма вычислений можно составить следующую параллельную форму.

Вариант 1

Данные $a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_7\ a_8$

Ярус 1 $a_1a_2 \quad a_3a_4 \quad a_5a_6 \quad a_7a_8$

Ярус 2 $a_1a_2 + a_3a_4 \quad a_5a_6 + a_7a_8$

Ярус 3 $(a_1a_2 + a_3a_4)(a_5a_6 + a_7a_8)$

Высота составленной параллельной формы равна трем, ширина – четырем. Она может быть реализована на параллельной системе, имеющей четыре процессора, способных выполнить операции умножения и сложения. Сначала выполняются четыре операции умножения, соответствующие ярусу 1, затем две операции сложения, соответствующие ярусу 2, и, наконец, одна операция умножения, соответствующая ярусу 3. Причем в рассмотренной форме все четыре процессора задействованы только на первом ярусе, затем работают два процессора и на третьем ярусе лишь один процессор загружен полезной работой, а все остальные простаивают.

Можно составить для данного алгоритма и другие параллельные формы, в которых высота будет больше, но при этом используемые процессоры загружены более интенсивно. Ниже приведены еще два варианта параллельной формы с высотой, равной четырем и пяти соответственно.

Вариант 2

Данные $a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_7\ a_8$

Ярус 1 $a_1a_2 \quad a_3a_4$

Ярус 2 $a_5a_6 \quad a_7a_8$

Ярус 3 $a_1a_2 + a_3a_4 \quad a_5a_6 + a_7a_8$

Ярус 4 $(a_1a_2 + a_3a_4)(a_5a_6 + a_7a_8)$

Вариант 3

Данные $a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_7\ a_8$

Ярус 1 $a_1a_2 \quad a_3a_4$

Ярус 2 $a_1a_2 + a_3a_4 \quad a_5a_6$

Ярус 3 a_7a_8

Ярус 4 $a_5a_6 + a_7a_8$

Ярус 5 $(a_1a_2 + a_3a_4)(a_5a_6 + a_7a_8)$

Ширина параллельных форм во втором и третьем вариантах равна двум. Обе эти параллельные формы могут быть реализованы на параллельной системе с двумя процессорами. Второй вариант является более удачным в плане загрузки процессоров, здесь все процессоры работают на всех ярусах, кроме последнего. В третьем варианте на третьем, четвертом и пятом ярусах один из двух задействованных процессоров простаивает.

□

Рассмотренный пример показывает некоторые проблемы, с которыми приходится сталкиваться при реализации алгоритмов на параллельных вычислительных системах. Одной из них является проблема загруженности процессоров.

Не менее важной проблемой реализации параллельных алгоритмов являются конфликты в памяти, когда несколько процессоров выбирают из памяти одну и ту же информацию. Конфликты в памяти можно ликвидировать за счет увеличения высоты, устанавливая фактически для процессоров очередность получения информации.

ПРИЛОЖЕНИЕ

Работа 1. Организация вычислений в среде MathCAD

Назначение программы MathCAD. Программа включает вычислительный процессор для проведения расчетов согласно введенным формулам, символьный процессор и текстовый редактор для ввода и редактирования формул и текста. Интеграция этих компонентов создает удобную вычислительную среду для разнообразных математических расчетов и документирования результатов работы.

В среде MathCAD можно выполнить следующие работы:

– ввод на компьютере математических выражений для дальнейших расчетов или создания документов, презентаций, Web-страниц;

– проведение математических расчетов;

– подготовка графиков по результатам расчетов;

– ввод исходных данных и вывод результатов в текстовые файлы или файлы с базами данных в различных форматах;

– подготовка отчетов работы в виде печатных документов;

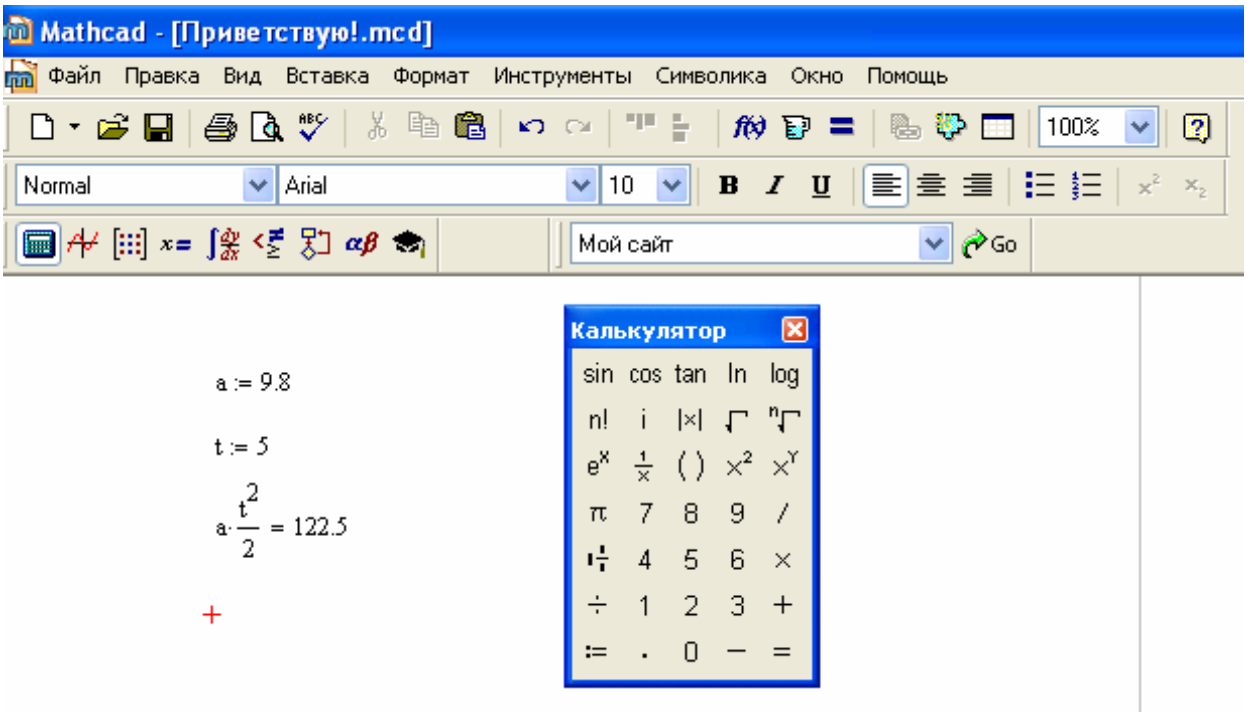
– подготовка Web-страниц и публикация результатов в Интернете;

– получение различной справочной информации из области математики.

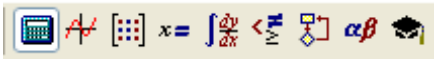
Интерфейс программы MathCAD. После того как программа запущена на исполнение, появляется основное окно приложения. Оно имеет ту же структуру, что и большинство приложений Windows. Сверху вниз располагаются заголовок окна, строка меню, панели инструментов (стандартная и форматирования) и рабочая область документа (worksheet). Новый документ создается автоматически при запуске MathCAD. В самой нижней части окна находится строка состояния. Не забывая о сходстве редактора MathCAD с обычными текстовыми редакторами, вы интуитивно поймете назначение большинства кнопок на панелях инструментов.

Панели инструментов служат для быстрого (в один щелчок мыши) выполнения наиболее часто применяемых команд. Все действия, которые можно выполнить с помощью панелей инструментов, доступны и через верхнее меню. Наибольший интерес представляет панель математических инструментов «Математика». Панель Math (Математика) предназначена для вызова на экран девяти служебных панелей, с помощью которых соб-

ственно и происходит вставка математических действий и операций в документы.

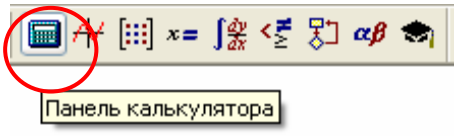


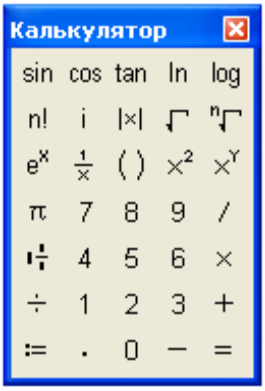
Панель инструментов «Математика» – основная панель программы, которая содержит несколько наборов инструментов, сгруппированных по выполняемым функциям и оформленных в виде всплывающих панелей:



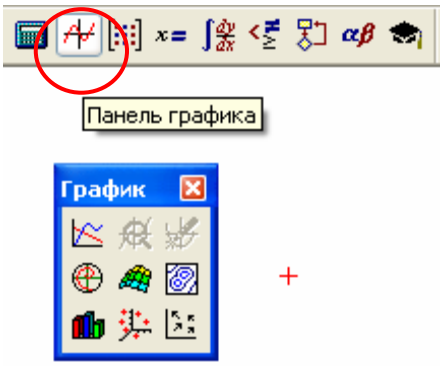
Перечислим назначение математических панелей.

Панель калькулятора – калькулятор, производит арифметические действия и простые вычисления

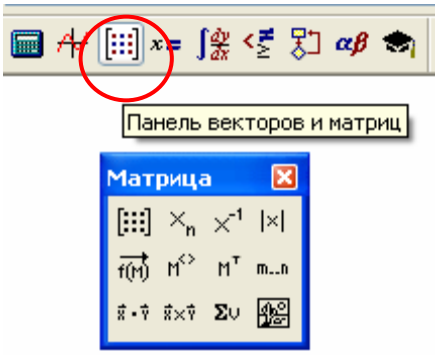




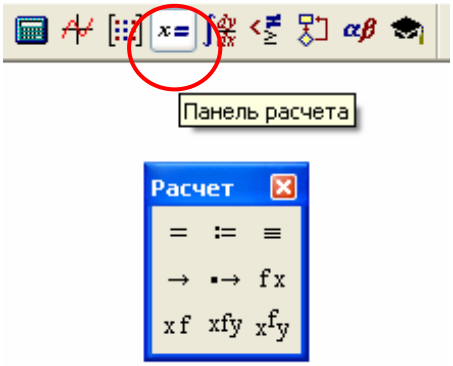
Панель графика – выбор типа графика



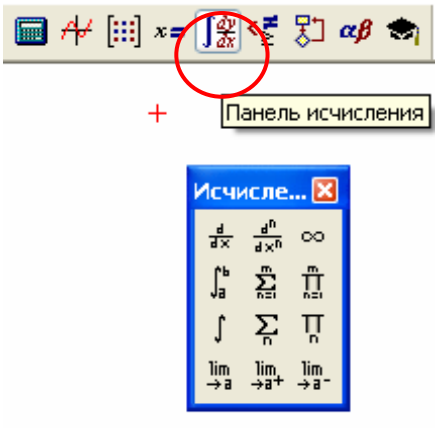
Панель векторов и матриц – действия с матрицами



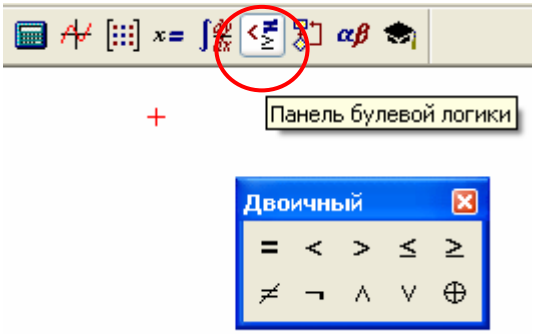
Панель расчетов – выполняемые действия и операторы



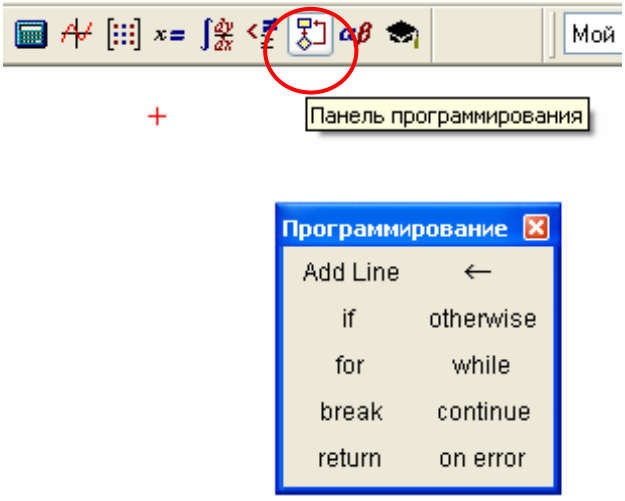
Панель исчисления – дифференцирование, интегрирование



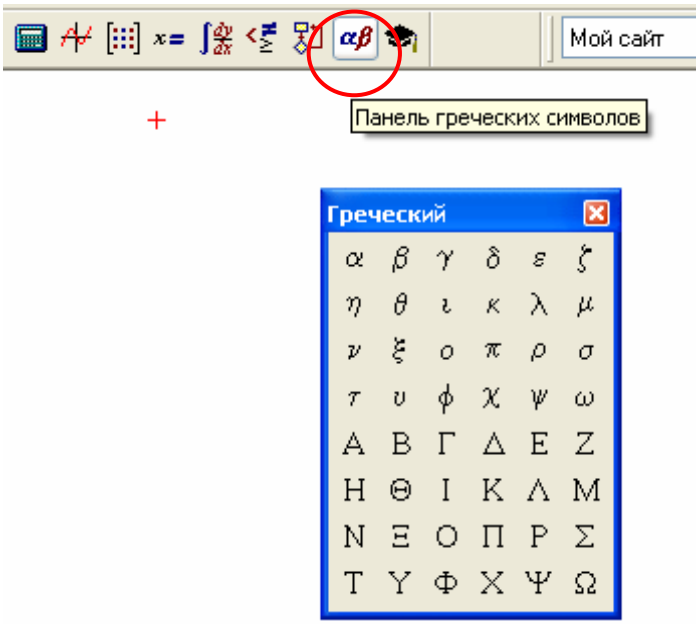
Панель булевой алгебры – булевы операции



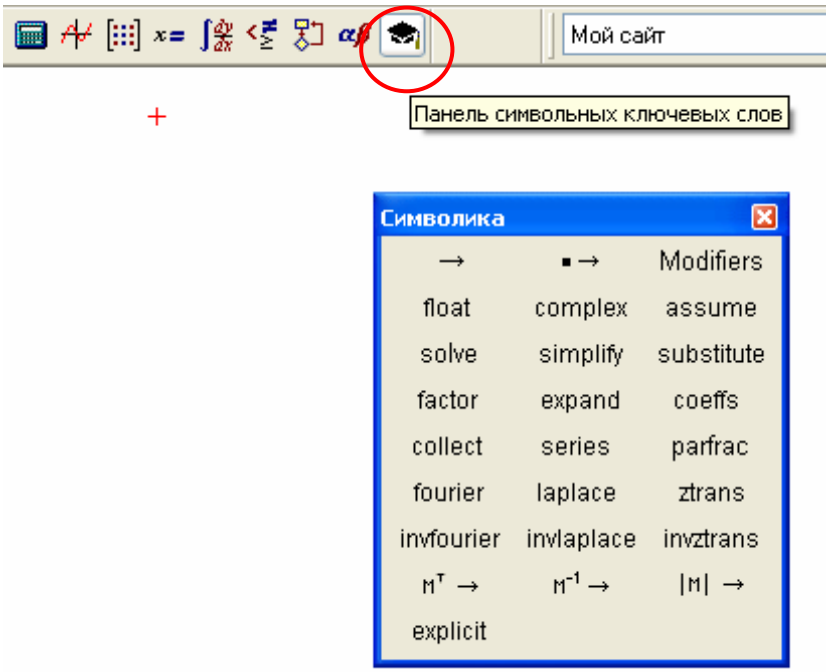
Панель программирования – команды для составления программ



Панель греческих символов – заглавные и прописные символы греческого алфавита



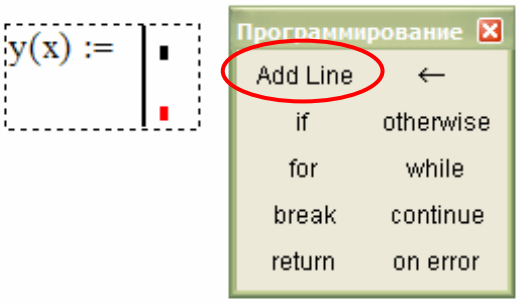
Панель символьных ключевых слов – команды, осуществляющие расчет и преобразование



Программирование в программе MathCAD.

Составление программы **Add Line**

Оператор добавляет строчку в программу.



Оператор присваивания ←

Оператор присваивает значение.

y(x) := a ← 3.2

ВСЕГДА (!!!) проверяйте текущий результат вычислений:

y(1.6) = 3.2

y(2168900) = 3.2

Заполним строки программы:

$$y(x) := \left| \begin{array}{l} a \leftarrow 3.2 \\ b \leftarrow x + a \end{array} \right.$$

Проверим результат вычислений:

$$y(1.6) = 4.8$$

Оператор условный **if**

Оператор задает условный переход:

$$y(x) := \left| \begin{array}{l} a \leftarrow 3.2 \\ b \leftarrow x + a \quad \text{if} \quad x > 2 \end{array} \right.$$

Проверим результат вычислений:

$$y(1) = 3.2 \qquad y(3) = 6.2$$

Оператор условный **otherwise**

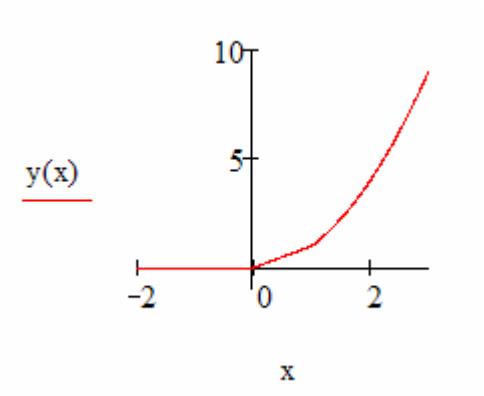
Оператор задает в программе переход для тех условий, которые не объявлены оператором *if:*

$$y(x) := \left| \begin{array}{ll} x^2 & \text{if} \quad x \geq 1 \\ x & \text{if} \quad 0 < x < 1 \\ 0 & \text{otherwise} \end{array} \right.$$

Проверим результат вычислений:

$$y(-5) = 0 \quad y(0) = 0 \quad y(0.5) = 0.5 \quad y(1) = 1 \quad y(3) = 9$$

График заданной функции:



Оператор цикла **while**

Оператор формирует цикл, выбирает нужный индекс и производит вычисление по выбранному индексу.

Поставим задачу так, что нужно найти первое значение, превышающее 3 для выражения:

$$i := 0..1000 \quad w_i := \sqrt[5]{i+1}$$

Зададим критерий:

$$R := 3$$

Создадим программу:

$$ww := \left| \begin{array}{l} i \leftarrow 14 \\ \text{while } w_i \le R \\ \quad i \leftarrow i + 1 \\ \begin{pmatrix} i \\ w_i \end{pmatrix} \end{array} \right.$$

Проверим результат вычислений:

$$ww = \begin{pmatrix} 243 \\ 3.002 \end{pmatrix}$$

Оператор цикла **for**

Оператор формирует цикл, листает индекс и проводит вычисления на каждом шаге.

Вычислим значения табличной функции:

$$y := \left| \begin{array}{l} i \leftarrow 1 \\ \text{for } x \in 1, 1.1..2 \\ \quad \left| \begin{array}{l} YYY_i \leftarrow \sin\left(x^2\right) \\ i \leftarrow i + 1 \end{array} \right. \\ YYY \end{array} \right.$$

Результат вычислений:

$y =$

	0
0	0
1	0.841
2	0.936
3	0.991
4	0.993
5	0.925
6	0.778
7	0.549
8	0.249
9	-0.098
10	-0.451
11	-0.757

Работа 2. Аппроксимация

Задача. Получена экспериментальная зависимость концентрации носителей заряда в полупроводнике от величины поверхностного сопротивления для электронного типа проводимости. Оценить концентрацию носителей заряда при поверхностном сопротивлении 0,0009 Ом*см.

P, Ом*см	0,003	0,005	0,01	0,02	0,03	0,04
N, $см^{-1}$	$2,41*10^{19}$	$1,27*10^{19}$	$4,76*10^{18}$	$1,51*10^{18}$	$7,22*10^{17}$	$4,25*10^{17}$

P, Ом*см	0,05	0,06	0,07
N, $см^{-1}$	$2,85*10^{17}$	$2,08*10^{17}$	$1,61*10^{17}$

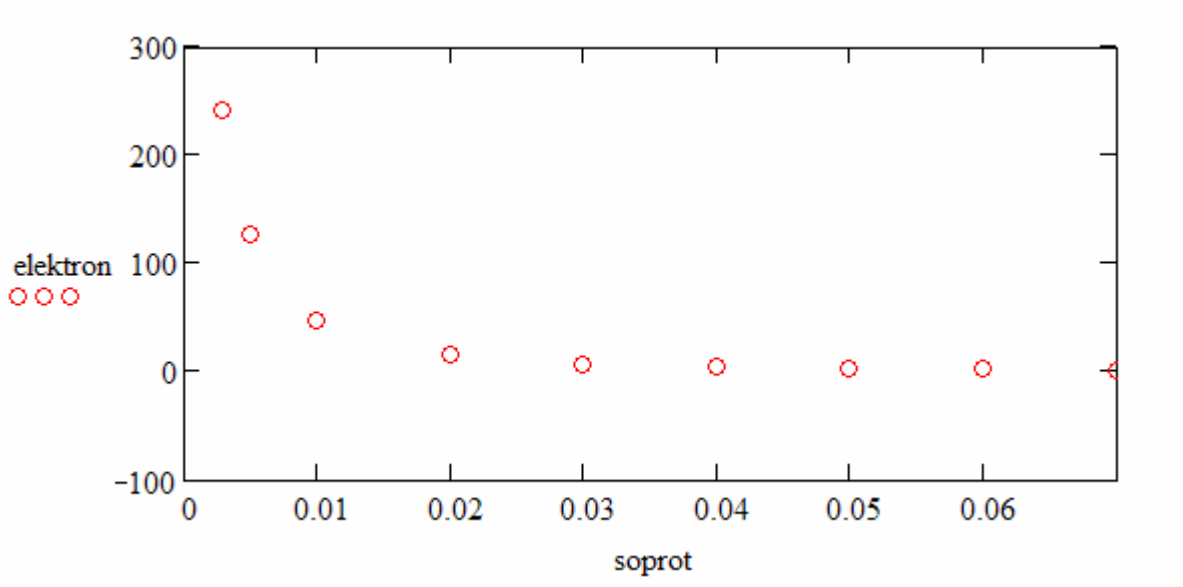
Решение.

Запишем исходные данные в виде матриц (векторов):

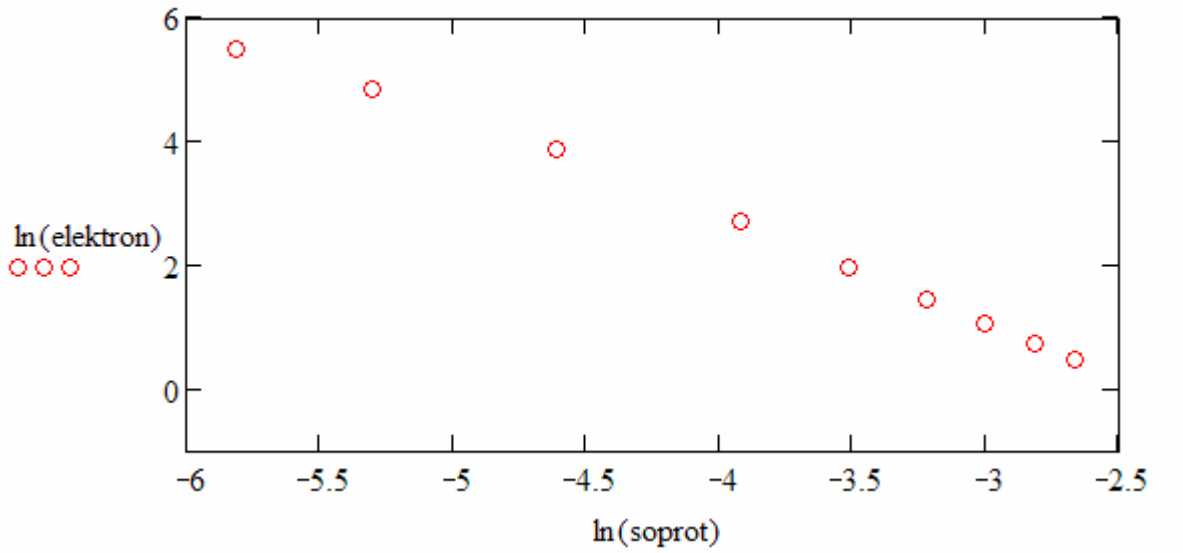
$$soprot := \begin{pmatrix} 0.003 \\ 0.005 \\ 0.01 \\ 0.02 \\ 0.03 \\ 0.04 \\ 0.05 \\ 0.06 \\ 0.07 \end{pmatrix} \qquad elektron := \begin{pmatrix} 241 \\ 127 \\ 47.6 \\ 15.1 \\ 7.22 \\ 4.25 \\ 2.85 \\ 2.08 \\ 1.61 \end{pmatrix}$$

В представлении данных к расчету проигнорирован порядок значений концентрации, такой подход повышает точность компьютерных вычислений.

Построим график исходной таблично заданной функции:



Поскольку функция имеет структуру экспоненты, перестроим ее график в логарифмических координатах:



В логарифмических координатах экспоненциальная функция представляется прямой линией, что позволяет свести аппроксимацию функции к линейной регрессии.

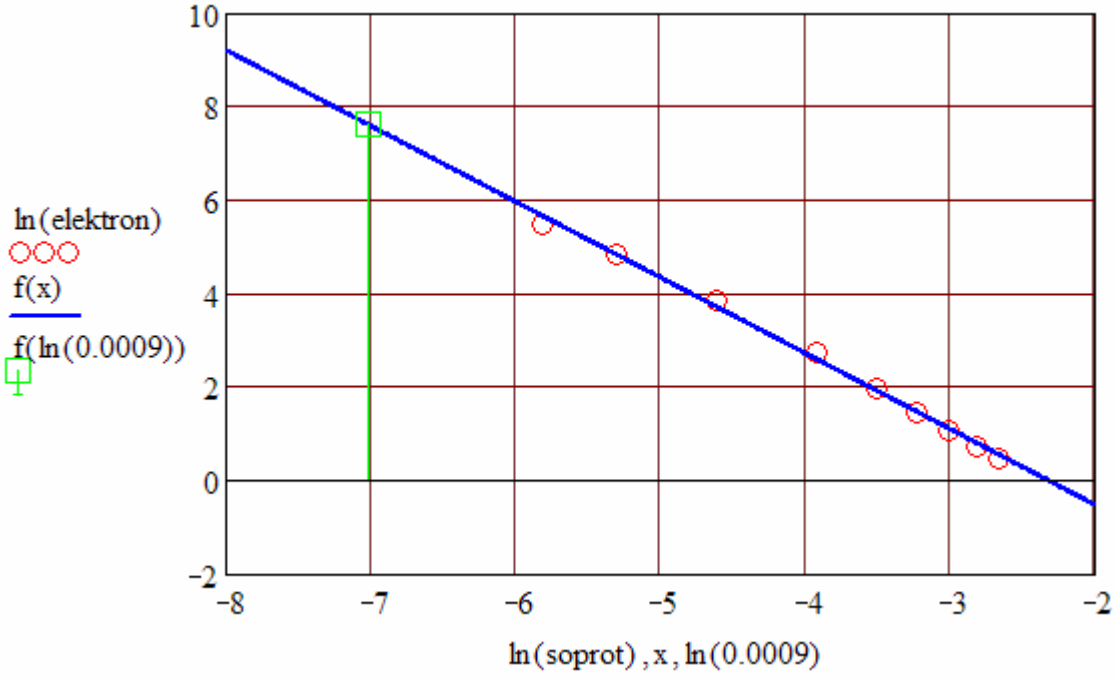
$$nomer := 0, 1 .. 8$$

$$z := regress(ln(soprot), ln(elektron), 8)$$

$$slope(ln(soprot), ln(elektron)) = -1.621$$

$$intercept(ln(soprot), ln(elektron)) = -3.763$$

$$f(x) := intercept(ln(soprot), ln(elektron)) + x \cdot slope(ln(soprot), ln(elektron))$$



$$f(\ln(0.0009)) = 7.608$$

$$otvet := e^{f(\ln(0.0009))} \cdot 10^{17}$$

$$otvet = 2.014 \times 10^{20}$$

□

Работа 3. Нелинейные уравнения

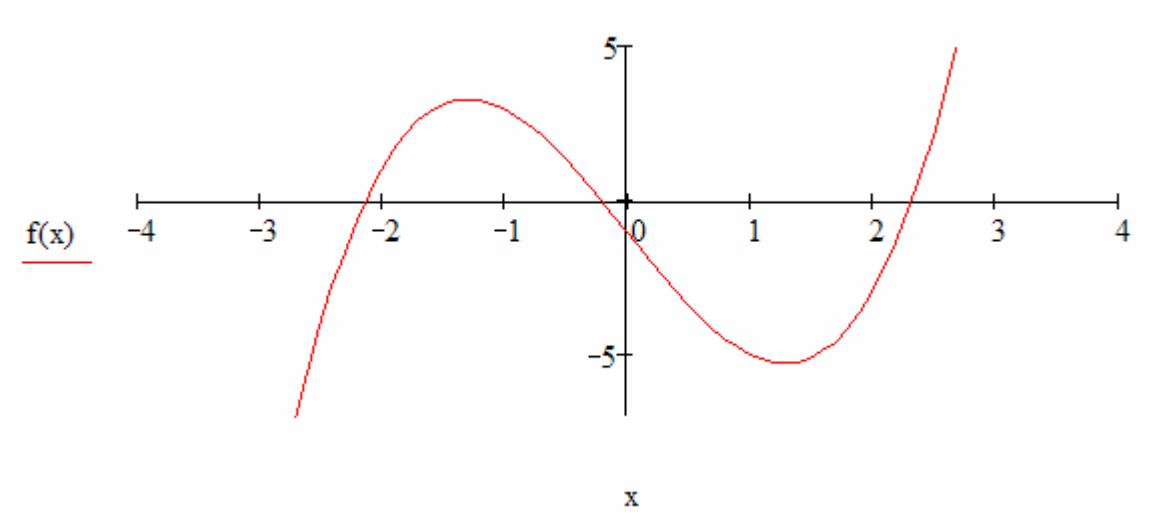
Задача: решить алгебраическое уравнение:

$$x^3 - 5 \cdot x - 1 = 0$$

Решение.

Этап первый: отделение корней.

Выполним отделение корней графически:



Вывод: уравнение имеет 3 корня.

Этап второй: уточнение корней.

Начальное приближение корня:

$x := 0$

Решение уравнения:

$\text{root}\left(x^3 - 5 \cdot x - 1, x\right) = -0.202$

Другой вариант записи:

$f(x) := x^3 - 5 \cdot x - 1 \qquad \text{root}(f(x), x) = -0.202$

Функция Root(f(x0),x0), где x0 – начальное приближение, возвращает такое значение x, при котором функция f(x) равна нулю.

Уточним другие корни:

$x := 2 \qquad \text{root}\left(x^3 - 5 \cdot x - 1, x\right) = 2.33$

$x := -2 \qquad \text{root}\left(x^3 - 5 \cdot x - 1, x\right) = -2.128$

□

Заключение

Небольшой вводный курс по вычислительным методам в компьютерном инжиниринге не может охватить всю проблематику быстро развивающейся компьютерной отрасли. Далее следует углубить свои знания в области математики и программирования, совершенствовать навыки решения инженерных задач в интерфейсах распространенных программных сред, можно перейти к написанию прикладных программ для решения инженерных и проектных задач.

Учебное издание

Огородникова Ольга Михайловна

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ
В КОМПЬЮТЕРНОМ ИНЖИНИРИНГЕ

Редакторы *Н.В.Рощина, Л.С.Гудкова*

Компьютерная верстка *К.Т.Горчаковой*

Подписано в печать 29.04.2013 Формат 60x90 1/16
Бумага писчая Плоская печать Усл. печ. л. 8,25
Уч.-изд. л. 7,0 Тираж 75 экз. Заказ № 694

Издательство Уральского университета
Редакционно-издательский отдел ИПЦ УрФУ
620049, Екатеринбург, ул. С.Ковалевской, 5
Тел.: +7(343) 375-48-25
rio@mail.ustu.ru

Отпечатано в Издательско-полиграфическом центре УрФУ
620000, Екатеринбург, ул. Тургенева, 4
Тел.: +7(343) 350-56-64
pres-urfu@mail.ru